IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of :

Taketo HEISHI et al. :

Serial No. NEW : **Attn: APPLICATION BRANCH**

Filed January 21, 2004 : Attorney Docket No. 2004_0070A

COMPILER APPARATUS AND
COMPILATION METHOD

THE COMMISSIONER IS AUTHORIZED TO CHARGE ANY DEFICIENCY IN THE FEES FOR THIS PAPER TO DEPOSIT ACCOUNT NO. 23-0975

CLAIM OF PRIORITY UNDER 35 USC 119

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Applicants in the above-entitled application hereby claim the date of priority under the International Convention of Japanese Patent Application No. 2003-019365, filed January 28, 2003, as acknowledged in the Declaration of this application.

A certified copy of said Japanese Patent Application is submitted herewith.

Respectfully submitted,

Taketo HEISHI et al.

By _____
Michael S. Huppert
Registration No. 40,268
Attorney for Applicants

MSH/kjf
Washington, D.C. 20006-1021
Telephone (202) 721-8200
Facsimile (202) 721-8250
January 21, 2004

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年　１月２８日

出願番号
Application Number: 特願２００３－０１９３６５

[ST. 10/C]: [ＪＰ２００３－０１９３６５]

出願人
Applicant(s): 松下電器産業株式会社

２００３年１０月１０日

特許庁長官
Commissioner,
Japan Patent Office

今井康夫

出証番号　出証特２００３－３０８３７９８

【書類名】　特許願
【整理番号】　5037740129
【あて先】　特許庁長官殿
【国際特許分類】　G06F　9/45
【発明者】
【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内
【氏名】　瓶子　岳人
【発明者】
【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内
【氏名】　小川　一
【発明者】
【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内
【氏名】　谷　丈暢
【発明者】
【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内
【氏名】　笹川　幸宏
【特許出願人】
【識別番号】　000005821
【氏名又は名称】　松下電器産業株式会社
【代理人】
【識別番号】　100109210
【弁理士】
【氏名又は名称】　新居　広守
【電話番号】　06-4806-7530

【手数料の表示】

【予納台帳番号】　049515

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　0213583

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　コンパイラ装置およびコンパイル方法

【特許請求の範囲】

【請求項１】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令とのハミング距離が小さくなる命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置し、前記中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段と

を備えることを特徴とするコンパイラ装置。

【請求項２】　前記最適化手段は、直前のサイクルの同一の命令発行ユニットに対応する位置に配置された命令のオペコードとのハミング距離が小さくなる命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置することを特徴とする請求項１に記載のコンパイラ装置。

【請求項３】　前記最適化手段は、直前のサイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタ番号とのハミング距離が小さくなる命令を優先して、前記複数の命令発行ユニットの各々に対応する位置に当該命令を配置することを特徴とする請求項１に記載のコンパイラ装置。

【請求項４】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタと同一のレジスタをアクセスする命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置し、前記中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段と

を備えることを特徴とするコンパイラ装置。

【請求項５】　前記最適化手段は、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタのレジスタ番号と近い番号のレジスタをアクセスする命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置することを特徴とする請求項４に記載のコンパイラ装置。

【請求項６】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、前記複数の命令発行ユニットの各々には、あらかじめ優先的に発行される命令が規定されており、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットの各々で優先発行される命令を優先して、前記複数の命令発行ユニットの各々に対応する位置に当該命令を配置し、前記中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段と

を備えることを特徴とするコンパイラ装置。

【請求項７】　ソースプログラムを、並列処理可能な複数の実行ユニットと

、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットのうち特定の命令発行ユニットに対応する位置には命令が配置されないように、前記複数の命令発行ユニットの各々の対応する位置に命令を配置し、前記中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段と

を備えることを特徴とするコンパイラ装置。

【請求項８】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記複数の命令発行ユニットにそれぞれ対応する複数の命令の配置位置について、同一数の命令の未配置位置があらかじめ定められた命令サイクル数以上連続する区間を検出する区間検出手段と、

前記区間の直前に、前記命令の未配置位置に対応する命令発行ユニットの動作を停止させるための命令を挿入する第１の命令挿入手段と、

当該命令が挿入された前記中間コードを機械語命令に変換するコード生成手段と

を備えることを特徴とするコンパイラ装置。

【請求項９】　さらに、前記区間の直後に、前記命令の未配置位置に対応する命令発行ユニットの動作を復帰させるための命令を挿入する第２の命令挿入手

段

を備えることを特徴とする請求項８に記載のコンパイラ装置。

【請求項１０】　前記プロセッサは、前記複数の命令発行ユニットの利用状態を示す値を保持するプログラム状態レジスタを有し、

前記第１の命令挿入手段で挿入される命令は、前記プログラム状態レジスタに前記複数の命令発行ユニットの利用状態を示す値を書き込むことを特徴とする請求項８または９に記載のコンパイラ装置。

【請求項１１】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記ソースプログラムは、前記プロセッサが使用する命令発行ユニットの個数を指定可能な個数指定情報を含み、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、前記個数指定情報で指定された個数の命令発行ユニットのみを動作させるように命令を配置し、前記中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段と

を備えることを特徴とするコンパイラ装置。

【請求項１２】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記プロセッサが使用する命令発行ユニットの個数を受付ける受付け手段と、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、前記受付け手段が受付けた前記個数の命令発行ユニットのみを動作させるように命令を配置し、前記中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段と

を備えることを特徴とするコンパイラ装置。

【請求項１３】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令とのハミング距離が小さくなるように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行ない、前記中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段と

を備えることを特徴とするコンパイラ装置。

【請求項１４】　前記最適化手段は、直前の命令サイクルの複数の命令発行ユニットに対応する位置にそれぞれ配置された複数の命令とのハミング距離の合計値が小さくなるように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行なうことを特徴とする請求項１３に記載のコンパイラ装置。

【請求項１５】　前記最適化手段は、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のオペコードとのハミング距離が小さくなるように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行なうことを特徴とする請求項１３に記載のコンパイラ装置。

【請求項１６】　前記最適化手段は、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタ番号とのハミング距離が小

さくなるように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行なうことを特徴とする請求項１３に記載のコンパイラ装置。

【請求項１７】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、各命令発行ユニットにおいて、直前の命令サイクルの命令のレジスタと同一のレジスタのアクセスが最も多く連続するように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行ない、前記中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段と

を備えることを特徴とするコンパイラ装置。

【請求項１８】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、前記複数の命令発行ユニットの各々には、あらかじめ優先的に発行される命令群が規定されており、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットの各々で優先的に発行される命令群に最も多く合致するように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行ない、前記中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段と

を備えることを特徴とするコンパイラ装置。

【請求項１９】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットに対応する位置に命令を配置する命令スケジューリング手段と、

前記命令スケジューリング手段でスケジュールされた各命令で使用される変数について、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタ番号とのハミング距離が小さくなるようなレジスタを優先して、当該変数に割付けるレジスタ割付手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段と

を備えることを特徴とするコンパイラ装置。

【請求項２０】　前記レジスタ割付手段は、スケジュールされた各命令で使用される変数について、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令と同一のレジスタを優先して、当該変数に割付けることを特徴とする請求項１９に記載のコンパイラ装置。

【請求項２１】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、複数のあらかじめ

定められた方法の各々に従い、前記複数の命令発行ユニットの各々の対応する位置に命令を配置し、生成された複数の配置命令のうち、プログラム実行時に最も消費電力が小さくなると予想される配置命令を選択し、前期中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段と

を備えることを特徴とするコンパイラ装置。

【請求項２２】　前記最適化手段は、

前記中間コードに対応する命令の依存関係を崩すことなく、複数のあらかじめ定められた配置方法の各々に従い、前記複数の命令発行ユニットに対応する位置に命令を配置する命令スケジューリング部と、

前記命令スケジューリング手段でスケジュールされた各命令で使用される変数について、複数のあらかじめ定められた割付方法の各々に従い、レジスタを前記変数に割付けるレジスタ割付部とを有し、

前記命令スケジューリング部での配置方法と、前記レジスタ割付部での割付方法とを、プログラム実行時に最も消費電力が小さくなると予想される命令の配置が得られるように、バックトラック方式で探索する

ことを特徴とする、請求項２１に記載のコンパイラ装置。

【請求項２３】　前記最適化手段は、複数のあらかじめ定められた再配置方法の各々に従い、前記複数の命令発行ユニットに対応する位置に、前記レジスタ割付部で変数にレジスタが割付けられた命令を、再配置する命令再スケジューリング部をさらに有し、

前記命令スケジューリング部での配置方法と、前記レジスタ割付部での割付方法と、前記命令再スケジューリング部での再配置方法とを、プログラム実行時に最も消費電力が小さくなると予想される命令の配置が得られるように、バックトラック方式で探索する

ことを特徴とする、請求項２２に記載のコンパイラ装置。

【請求項２４】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法で

あって、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令とのハミング距離が小さくなる命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするコンパイル方法。

【請求項２５】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタと同一のレジスタをアクセスする命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするコンパイル方法。

【請求項２６】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、前記複数の命令発行ユニットの各々には、あらかじめ優先的に発行される命令が規定されており、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットの各々で優先発行される命令を優先して、前記複数の命令発行ユニットの各々に対応する位置に当該命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするコンパイル方法。

【請求項２７】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットのうち特定の命令発行ユニットに対応する位置には命令が配置されないように、前記複数の命令発行ユニットの各々の対応する位置に命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするコンパイル方法。

【請求項２８】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記複数の命令発行ユニットにそれぞれ対応する複数の命令の配置位置について、同一数の命令の未配置位置があらかじめ定められた命令サイクル数以上連続する区間を検出する区間検出ステップと、

前記区間の直前に、前記命令の未配置位置に対応する命令発行ユニットの動作を停止させるための命令を挿入する第１の命令挿入ステップと、

当該命令が挿入された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするコンパイル方法。

【請求項２９】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、

前記ソースプログラムは、前記プロセッサが使用する命令発行ユニットの個数を指定可能な個数指定情報を含み、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記個数指定情報で指定された個数の命令発行ユニットのみを動作させるように命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするコンパイル方法。

【請求項３０】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、

前記プロセッサが使用する命令発行ユニットの個数を受付ける受付けステップと、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記個数の命令発行ユニットのみを動作させるように命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするコンパイル方法。

【請求項３１】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令とのハミング距離が小さくなるように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行ない、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするコンパイル方法。

【請求項３２】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、各命令発行ユニッ

トにおいて、直前の命令サイクルの命令のレジスタと同一のレジスタのアクセスが最も多く連続するように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行ない、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするコンパイル方法。

【請求項３３】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、前記複数の命令発行ユニットの各々には、あらかじめ優先的に発行される命令群が規定されており、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットの各々で優先的に発行される命令群に最も多く合致するように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行ない、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするコンパイル方法。

【請求項３４】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットに対応する位置に命令を配置する命令スケジューリングステップと、

スケジュールされた各命令で使用される変数について、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタ番号とのハミング距離が小さくなるようなレジスタを優先して、当該変数に割付けるレジスタ割付ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするコンパイル方法。

【請求項３５】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、複数のあらかじめ定められた方法の各々に従い、前記複数の命令発行ユニットの各々の対応する位置に命令を配置し、生成された複数の配置命令のうち、プログラム実行時に最も消費電力が小さくなると予想される配置命令を選択し、前期中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするコンパイル方法。

【請求項３６】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイク

ルの同一の命令発行ユニットに対応する位置に配置された命令とのハミング距離が小さくなる命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするプログラム。

【請求項３７】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタと同一のレジスタをアクセスする命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするプログラム。

【請求項３８】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、前記複数の命令発行ユニットの各々には、あらかじめ優先的に発行される命令が規定されており、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットの各々で優先発行される命令を優先して、前記複数の命令発行ユニットの各々に対応する位置に当該命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするプログラム。

【請求項３９】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットのうち特定の命令発行ユニットに対応する位置には命令が配置されないように、前記複数の命令発行ユニットの各々の対応する位置に命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするプログラム。

【請求項４０】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記複数の命令発行ユニットにそれぞれ対応する複数の命令の配置位置につい

て、同一数の命令の未配置位置があらかじめ定められた命令サイクル数以上連続する区間を検出する区間検出ステップと、

前記区間の直前に、前記命令の未配置位置に対応する命令発行ユニットの動作を停止させるための命令を挿入する第１の命令挿入ステップと、

当該命令が挿入された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするプログラム。

【請求項４１】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記ソースプログラムは、前記プロセッサが使用する命令発行ユニットの個数を指定可能な個数指定情報を含み、

前記プログラムは、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記個数指定情報で指定された個数の命令発行ユニットのみを動作させるように命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするプログラム。

【請求項４２】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、

前記プロセッサが使用する命令発行ユニットの個数を受付ける受付けステップと、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記個数の命令発行ユニットのみを動作させるように命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするプログラム。

【請求項４３】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令とのハミング距離が小さくなるように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行ない、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするプログラム。

【請求項４４】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、各命令発行ユニットにおいて、直前の命令サイクルの命令のレジスタと同一のレジスタのアクセスが最も多く連続するように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行ない、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするプログラム。

【請求項４５】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、前記複数の命令発行ユニットの各々には、あらかじめ優先的に発行される命令群が規定されており、

前記プログラムは、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットの各々で優先的に発行される命令群に最も多く合致するように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行ない、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするプログラム。

【請求項４６】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットに対応する位置に命令を配置する命令スケジューリングステップと、

スケジュールされた各命令で使用される変数について、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタ番号とのハミング距離が小さくなるようなレジスタを優先して、当該変数に割付けるレジスタ割付ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするプログラム。

【請求項４７】　ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、複数のあらかじめ定められた方法の各々に従い、前記複数の命令発行ユニットの各々の対応する位置に命令を配置し、生成された複数の配置命令のうち、プログラム実行時に最も消費電力が小さくなると予想される配置命令を選択し、前期中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするプログラム。

【請求項４８】　並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語コードが記録されたコンピュータ読取可能な記録媒体であって

前記機械語コードは、

前記複数の命令発行ユニットのいずれかで、あらかじめ定められた命令サイクル数以上命令が実行されない期間において当該命令発行ユニットの動作を停止させる第１の命令

を備えることを特徴とするコンピュータ読取可能な記録媒体。

【請求項４９】　前記機械語コードは、さらに、

前記期間の経過直後に当該命令発行ユニットの動作を復帰させるための第２の命令

を備えることを特徴とする請求項４８に記載のコンピュータ読取可能な記録媒体。

【請求項５０】　前記プロセッサは、前記複数の命令発行ユニットの利用状態を示す値を保持するプログラム状態レジスタを有し、

前記第１の命令は、前記プログラム状態レジスタに前記複数の命令発行ユニットの利用状態を示す値を書き込むことを特徴とする請求項４８または４９に記載のコンピュータ読取可能な記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、Ｃ／Ｃ＋＋言語等の高級言語で記述されたソースプログラムを機械語プログラムに変換するコンパイラに関し、特に、プログラム実行時の消費電力が少ない機械語プログラムを出力可能なコンパイラに関する。

【０００２】

【従来の技術】

近年普及している携帯電話、情報携帯端末等の携帯型の情報処理装置では、消費電力の低減が要求されている。このため、このような情報処理装置で利用されるプロセッサが備える高い機能を有効に引き出しつつ、低消費電力で動作可能な機械語命令を生成可能なコンパイラが求められている。

【０００３】

従来のコンパイラとして、命令の実行順序を変更することによりプロセッサの消費電力の低減を図った命令列最適化装置がある（たとえば、特許文献１参照。）。

【０００４】

この命令列最適化装置では、命令の依存関係に影響を与えることなく、命令のビットパターン間のハミング距離を低減させるように命令の配置変更を行なう。これにより、プロセッサの消費電力を低減させることができる命令列の最適化を行なうことができる。

【０００５】

【特許文献１】

特開平８－１０１７７７号公報

【０００６】

【発明が解決しようとする課題】

しかしながら、従来の命令列最適化装置では、並列処理可能なプロセッサを前提としていない。このため、従来の最適化処理を並列処理可能なプロセッサにそのまま適用しても、最適な命令列の最適化を得ることができないという問題がある。

【０００７】

そこで、本発明は、このような状況に鑑みてなされたものであり、並列処理可能なプロセッサを低消費電力で動作させることができる命令列を生成可能なコンパイラを提供することを目的とする。

【０００８】

【課題を解決するための手段】

上記目的を達成するために、本発明に係るコンパイラ装置は、ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、前記ソースプログラムを構文解析するパーサー手段と、解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、前記中間コードに対応する命令の依存関係を崩すこ

となく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令とのハミング距離が小さくなる命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置し、前記中間コードを最適化する最適化手段と、最適化された前記中間コードを機械語命令に変換するコード生成手段とを備えることを特徴とする。

【０００９】

これにより、各命令発行ユニットで実行される命令のビットパターンの変化を抑えることができるので、プロセッサの命令レジスタに保持される値のビット変化が小さく、プロセッサを低消費電力で動作させることができる命令列が生成される。

【００１０】

また、本発明に係るコンパイラ装置は、ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、前記ソースプログラムを構文解析するパーサー手段と、解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタと同一のレジスタをアクセスする命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置し、前記中間コードを最適化する最適化手段と、最適化された前記中間コードを機械語命令に変換するコード生成手段とを備えることを特徴とする。

【００１１】

これにより、同一のレジスタへのアクセスが連続し、レジスタを選択するための制御信号の変化が少なくなり、プロセッサを低消費電力で動作させることができる命令列が生成される。

【００１２】

さらに、本発明に係るコンパイラ装置は、ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行

する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、前記複数の命令発行ユニットの各々には、あらかじめ優先的に発行される命令が規定されており、前記ソースプログラムを構文解析するパーサー手段と、解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットの各々で優先発行される命令を優先して、前記複数の命令発行ユニットの各々に対応する位置に当該命令を配置し、前記中間コードを最適化する最適化手段と、最適化された前記中間コードを機械語命令に変換するコード生成手段とを備えることを特徴とする。

【００１３】

これにより、同じ命令発行ユニットで優先発行される命令として、プロセッサの同じ構成要素を利用する命令を割り当てれば、同一の命令発行ユニットでは、同じ構成要素を利用する命令が連続して実行されることになるため、プロセッサを低消費電力で動作させることができる命令列が生成される。

【００１４】

さらにまた、本発明に係るコンパイラ装置は、ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、前記ソースプログラムを構文解析するパーサー手段と、解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、前記複数の命令発行ユニットにそれぞれ対応する複数の命令の配置位置について、同一数の命令の未配置位置があらかじめ定められた命令サイクル数以上連続する区間を検出する区間検出手段と、前記区間の直前に、前記命令の未配置位置に対応する命令発行ユニットの動作を停止させるための命令を挿入する第１の命令挿入手段と、当該命令が挿入された前記中間コードを機械語命令に変換するコード生成手段とを備えることを特徴とする。

【００１５】

これにより、命令発行ユニットに対応する位置に命令が連続して配置されていない場合には、その間、その命令発行ユニットへの電力の供給を停止させること

ができため、プロセッサを低消費電力で動作させることができる命令列が生成される。

【００１６】

さらにまた、本発明に係るコンパイラ装置は、ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、前記ソースプログラムは、前記プロセッサが使用する命令発行ユニットの個数を指定可能な個数指定情報を含み、前記ソースプログラムを構文解析するパーサー手段と、解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、前記中間コードに対応する命令の依存関係を崩すことなく、前記個数指定情報で指定された個数の命令発行ユニットのみを動作させるように命令を配置し、前記中間コードを最適化する最適化手段と、最適化された前記中間コードを機械語命令に変換するコード生成手段とを備えることを特徴とする。

【００１７】

これにより、個数指定情報で指定された箇所の命令に関しては、命令が供給されない命令発行ユニットを発生させることができ、その命令発行ユニットへの電力の供給を停止させることができるため、プロセッサを低消費電力で動作させることができる命令列が生成される。

【００１８】

さらにまた、本発明に係るコンパイラ装置は、ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、前記プロセッサが使用する命令発行ユニットの個数を受付ける受付け手段と、前記ソースプログラムを構文解析するパーサー手段と、解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、前記中間コードに対応する命令の依存関係を崩すことなく、前記受付け手段が受付けた前記個数の命令発行ユニットのみを動作させるように命令を配置し、前記中間コードを最適化する最適化手段と、最適化された前記中間コード

を機械語命令に変換するコード生成手段とを備えることを特徴とする。

【００１９】

これにより、受付け手段で受付けた個数の命令発行ユニットのみを動作させ、その他の命令発行ユニットへの電力の供給を停止させることができるため、プロセッサを低消費電力で動作させることができる命令例が生成される。

【００２０】

なお、本発明は、このようなコンパイラ装置として実現することができるだけでなく、このようなプログラムに含まれる手段をステップとするコンパイル方法として実現したり、このような特徴的なコンパイラ用のプログラムまたはコンピュータ読取可能な記録媒体として実現したりすることもできる。そして、そのようなプログラムやデータファイルは、ＣＤ－ＲＯＭ（Compact Disk-Read Only Memory）等の記録媒体やインターネット等の伝送媒体を介して広く流通させることができるのは言うまでもない。

【００２１】

【発明の実施の形態】

以下、本発明に係るコンパイラの実施の形態について図面を用いて詳細に説明する。

【００２２】

本実施の形態におけるコンパイラは、Ｃ／Ｃ＋＋言語等の高級言語で記述されたソースプログラムを特定のプロセッサ（ターゲット）が実行できる機械語プログラムに翻訳するクロスコンパイラであり、プロセッサの消費電力を低減させることができるという特徴を有する。

【００２３】

[プロセッサ]

まず、本実施の形態におけるコンパイラの対象となるプロセッサの一例について、図１～図１１を用いて説明する。

【００２４】

本実施の形態におけるコンパイラの対象となるプロセッサは、例えば、通常のマイコンに比べて実行可能な命令の並列性が高く、複数の命令を並列して処理で

きるようにパイプライン方式が採用されている。

【００２５】

図１は、本実施の形態に係るプロセッサが解読実行する命令の構造を示す図である。

図１（ａ）～図１（ｄ）を参照して、本プロセッサの各命令は、３２ビットの固定長である。各命令の０ビット目は、並列実行境界情報を示す。並列実行境界情報が“１”のときは、その命令と後続命令との間に並列実行の境界が存在し、並列実行境界情報が“０”のときは、並列実行の境界が存在しないことになる。並列実行境界情報の利用方法については、後述する。

【００２６】

各命令の命令長から並列実行境界情報を除いた３１ビットの部分においてオペレーションを決定する。具体的には、フィールド“Ｏｐ１”、“Ｏｐ２”、“Ｏｐ３”および“Ｏｐ４”では、オペレーションの種類を示すオペコードを指定する。レジスタフィールド“Ｒｓ”、“Ｒｓ１”および“Ｒｓ２”では、ソースオペランドとなるレジスタのレジスタ番号を指定する。レジスタフィールド“Ｒｄ”では、ディスティネーションオペランドとなるレジスタのレジスタ番号を指定する。フィールド“Ｉｍｍ”では、演算用定数オペランドを指定する。フィールド“Ｄｉｓｐ”では、変位（ディスプレースメント）を指定する。

【００２７】

オペコードの先頭２ビット（３０および３１ビット目）は、オペレーションの種類（オペレーション群）を指定するために用いられる。その詳細については、後述する。

【００２８】

オペコードＯｐ２～Ｏｐ４は、１６ビット長のデータであるが、オペコードＯｐ１は２１ビット長のデータである。したがって、便宜的に、オペコードＯｐ１の前半部分（１６～３１ビット目）をオペコードＯｐ１－１と呼び、後半部分（１１～１５ビット目）をオペコードＯｐ１－２と呼ぶ。

【００２９】

図２は、本実施の形態に係るプロセッサの概略構成を示すブロック図である。

プロセッサ３０は、ＶＬＩＷ（Very Long Instruction Word）方式に従って記述された命令群（以下「パケット」と呼ぶ。）を記憶する命令メモリ４０と、命令供給発行部５０と、解読部６０と、実行部７０と、データメモリ１００とを含む。各部の詳細については後述する。

【００３０】

図３は、パケットの一例を示す図である。１パケットは、命令フェッチの単位であり、４命令から構成されるものと定義する。上述のように、１命令は３２ビット長である。このため、１パケットは、１２８（＝３２×４）ビット長である。

【００３１】

再度図２を参照して、命令供給発行部５０は、命令メモリ４０、解読部６０および実行部７０に接続され、実行部７０より供給されるＰＣ（プログラムカウンタ）の値に基づいて、命令メモリ４０よりパケットを受信し、解読部６０に並列して最高３つの命令を供給する。

【００３２】

解読部６０は、命令供給発行部５０および実行部７０に接続され、命令供給発行部５０から供給された命令を解読し、実行部７０に供給する。

実行部７０は、命令供給発行部５０、解読部６０およびデータメモリ１００に接続され、解読部６０より供給される解読結果に基づいて、必要に応じデータメモリ１００に記憶されたデータアクセスを行ない、命令に基づいた処理を実行する。また、処理が実行されるごとに、実行部７０はＰＣの値を１つずつインクリメントする。

【００３３】

命令供給発行部５０は、命令メモリ４０および実行部７０内の後述するＰＣ部に接続され、ＰＣ部に保持されたプログラムカウンタで示される命令メモリ４０のアドレスにアクセスし、命令メモリ４０よりパケットを受信する命令フェッチ部５２と、命令フェッチ部５２に接続され、パケットを一時的に保持する命令バッファ５４と、命令バッファ５４に接続され、パケットに含まれる命令を最大３つ保持する命令レジスタ部５６とを含む。

【００３４】

命令フェッチ部５２および命令メモリ４０は、ＩＡ（Instruction Address）バス４２およびＩＤ（Instruction Data）バス４４により接続されている。ＩＡバス４２は３２ビット幅であり、ＩＤバス４４は１２８ビット幅である。命令フェッチ部５２から命令メモリ４０へのアドレス供給は、ＩＡバス４２を介して行われる。命令メモリ４０から命令フェッチ部５２へのパケットの供給は、ＩＤバス４４を解して行われる。

【００３５】

命令レジスタ部５６は、各々命令バッファ５４に接続され、それぞれ１つの命令を保持する命令レジスタ５６ａ～５６ｃを含む。

解読部６０は、命令レジスタ部５６内の３つの命令レジスタ５６ａ～５６ｃに保持された命令の発行に関する制御を行う命令発行制御部６２と、命令発行制御部６２および命令レジスタ部５６に接続され、命令発行制御部６２の制御に基づいて、命令レジスタ部５６から供給される命令をデコードするデコード部６４とを含む。

【００３６】

デコード部６４は、命令レジスタ５６ａ～５６ｃにそれぞれ接続され、基本的に１サイクルに１つの命令を解読し、制御信号を出力する命令デコーダ６４ａ～６４ｃを含む。

【００３７】

実行部７０は、デコード部６４に接続され、デコード部６４内の３つの命令デコーダ６４ａ～６４ｃより出力される制御信号に基づいて、実行部７０内の後述する各構成要素を制御する実行制御部７２と、次に実行すべきパケットのアドレスを保持するＰＣ部７４と、３２個の３２ビットレジスタＲ０～Ｒ３１から構成されるレジスタファイル７６と、各々、ＳＩＭＤ（Single Instruction Multiple Data）型命令の演算を実行する算術論理・比較演算部７８ａ～７８ｃと、算術論理・比較演算部７８ａ～７８ｃと同様、ＳＩＭＤ型命令の実行が可能であり、ビット精度を落とさないように、最長で６５ビットで累算する乗算・積和演算部８０ａおよび８０ｂとを含む。

【００３８】

実行部７０は、さらに、各々、データの算術シフト（２の補数体系のシフト）または論理シフト（符号なしシフト）を実行するバレルシフタ８２ａ～８２ｂと、除算器８４と、データメモリ１００に接続され、データメモリ１００との間でデータの受渡しを行うオペランドアクセス部８８と、３２ビット幅のデータバス９０（Ｌ１バス、Ｒ１バス、Ｌ２バス、Ｒ２バス、Ｌ３バス、Ｒ３バス）と、３２ビット幅のデータバス９２（Ｄ１バス、Ｄ２バス、Ｄ３バス）とを含む。

【００３９】

レジスタファイル７６は、３２個の３２ビットレジスタＲ０～Ｒ３１を含む。Ｌ１バス、Ｒ１バス、Ｌ２バス、Ｒ２バス、Ｌ３バスおよびＲ３バスにデータを出力するレジスタファイル７６内のレジスタの選択は、実行制御部７２よりレジスタファイル７６に供給される制御信号ＣＬ１、ＣＲ１、ＣＬ２、ＣＲ２、ＣＬ３およびＣＲ３によりそれぞれ行なわれる。また、Ｄ１バス、Ｄ２バスおよびＤ３バスを流れるデータが書き込まれるレジスタの選択は、実行制御部７２よりレジスタファイル７６に供給される制御信号ＣＤ１、ＣＤ２およびＣＤ３によりそれぞれ行なわれる。

【００４０】

算術論理・比較演算部７８ａの２つの入力ポートは、Ｌ１バスおよびＲ１バスにそれぞれ接続され、その出力ポートは、Ｄ１バスに接続されている。算術論理・比較演算部７８ｂの２つの入力ポートは、Ｌ２バスおよびＲ２バスにそれぞれ接続され、その出力ポートは、Ｄ２バスに接続されている。算術論理・比較演算部７８ｃの２つの入力ポートは、Ｌ３バスおよびＲ３バスにそれぞれ接続され、その出力ポートは、Ｄ３バスに接続されている。

【００４１】

乗算・積和演算部８０ａの４つの入力ポートは、Ｌ１バス、Ｒ１バス、Ｌ２バスおよびＲ２バスにそれぞれ接続され、その２つの出力ポートは、Ｄ１バスおよびＤ２バスにそれぞれ接続されている。乗算・積和演算部８０ｂの４つの入力ポートは、Ｌ２バス、Ｒ２バス、Ｌ３バスおよびＲ３バスにそれぞれ接続され、その２つの出力ポートは、Ｄ２バスおよびＤ３バスにそれぞれ接続されている。

【００４２】

バレルシフタ８２ａの２つの入力ポートは、Ｌ１バスおよびＲ１バスにそれぞれ接続され、その出力ポートは、Ｄ１バスに接続されている。バレルシフタ８２ｂの２つの入力ポートは、Ｌ２バスおよびＲ２バスにそれぞれ接続され、その出力ポートは、Ｄ２バスに接続されている。バレルシフタ８２ｃの２つの入力ポートは、Ｌ３バスおよびＲ３バスにそれぞれ接続され、その出力ポートは、Ｄ３バスに接続されている。

【００４３】

除算器８４の２つの入力ポートは、Ｌ１バスおよびＲ１バスにそれぞれ接続され、その出力ポートは、Ｄ１バスに接続されている。

オペランドアクセス部８８およびデータメモリ１００は、ＯＡ（Operand Address）バス９６およびＯＤ（Operand Data）バス９４により接続されている。ＯＡバス９６およびＯＤバス９４はそれぞれ３２ビット幅である。また、オペランドアクセス部８８は、ＯＡバス９６を介してデータメモリ１００のアドレスを指定し、ＯＤバス９４を介して、当該アドレスのデータの読み書きを行なう。

【００４４】

また、オペランドアクセス部８８は、Ｄ１バス、Ｄ２バス、Ｄ３バス、Ｌ１バスおよびＲ１バスに接続され、いずれかのバスとの間でデータの受渡しを行なう。

【００４５】

プロセッサ３０は、３命令を並列実行可能であるが、後述するように、パイプライン動作のうち、並列実行可能な命令割り当てステージ、デコードステージ、実行ステージおよび書き込みステージからなる１組のパイプライン処理を実行させうる回路の集合を本明細書中で「スロット」と定義する。したがって、プロセッサ３０は、第１～第３スロットの３つのスロットを有する。命令レジスタ５６ａおよび命令デコーダ６４ａの組は第１スロット、命令レジスタ５６ｂおよび命令デコーダ６４ｂの組は第２スロット、命令レジスタ５６ｃおよび命令デコーダ６４ｃの組は第３スロットにそれぞれ属するものとする。

【００４６】

各スロットには、デフォルト論理と呼ばれる命令が割り当てられており、同一のスロットでは極力同一の命令が実行されるように、命令スケジューリングが行なわれる。例えば、第１スロットにはメモリアクセスに関する命令（デフォルト論理）、第２スロットには乗算に関するデフォルト論理、第３スロットにはその他のデフォルト論理が割り当てられている。なお、デフォルト論理は、図１を参照して説明したオペレーション群と一対一に対応している。すなわち、先頭２ビットが“０１”、“１０”および“１１”の命令は、それぞれ第１スロット、第２スロットおよび第３スロットのデフォルト論理である。

【００４７】

第１スロットのデフォルト論理としては、ｌｄ（ロード命令）、ｓｔ（ストア命令）などがある。第２スロットのデフォルト論理としては、ｍｕｌ１，ｍｕｌ２（乗算命令）などがある。第３スロットのデフォルト論理としては、ａｄｄ１，ａｄｄ２（加算命令）、ｓｕｂ１，ｓｕｂ２（減算命令）、ｍｏｖ１，ｍｏｖ２（レジスタ間の転送命令）などがある。

【００４８】

図４は、パケットに含まれる並列実行境界情報について説明するための図である。命令メモリ４０には、パケット１１２および１１４の順でパケットが記憶されているものとする。このうち、パケット１１２の命令２およびパケット１１４の命令５の並列実行境界情報は“１”であるものとし、それ以外の命令の並列実行境界情報は“０”であるものとする。

【００４９】

命令フェッチ部５２は、ＰＣ部７４のプログラムカウンタの値に基づいて、パケット１１２、パケット１１４の順でパケットを読み込み、順次、命令バッファ５４に供給する。実行部７０では並列実行境界情報が“１”までの命令が並列実行される。

【００５０】

図５は、パケットの並列実行境界情報に基づいて作成された並列実行される命令の実行単位の一例を示す図である。図４および図５を参照して、パケット１１２および１１４を並列実行境界情報が“１”の命令の部分で区切ると、実行単位

１２２～１２６が生成される。したがって、命令バッファ５４から命令レジスタ部５６には、実行単位１２２～１２６の順で命令が供給されることとなる。これらの、命令の供給に関する制御は、命令発行制御部６２が行なう。

【００５１】

命令デコーダ６４ａ～６４ｃは、命令レジスタ５６ａ～５６ｃに保持された命令のオペコードをそれぞれ解読し、制御信号を実行制御部７２に出力する。実行制御部７２は、命令デコーダ６４ａ～６４ｃでの解析結果に基づいて、実行部７０の構成要素の各種制御を行なう。

【００５２】

例えば、命令“ａｄｄｌ　Ｒ３，Ｒ０”について考える。この命令の意味は、レジスタＲ３の値とレジスタＲ０の値とを加算し、結果をレジスタＲ０に書き込むというものであるが、この場合、実行制御部７２は、以下のような制御を一例として行なう。実行制御部７２は、レジスタＲ３に保持された値をＬ１バスに出力するための制御信号ＣＬ１をレジスタファイル７６に供給する。また、実行制御部７２は、レジスタＲ０に保持された値をＲ１バスに出力するための制御信号ＣＲ１をレジスタファイル７６に供給する。

【００５３】

さらに、実行制御部７２は、Ｄ１バスを介して得られる実行結果をレジスタＲ０に書き込むための制御信号ＣＤ１をレジスタファイル７６に供給する。さらにまた、実行制御部７２は算術論理・比較演算部７８ａを制御し、Ｌ１バスおよびＬ２バスを介してレジスタＲ３およびＲ０の値を受け取り、加算した後、加算結果をＤ１バスを介してレジスタＲ０に書き込む。

【００５４】

図６は、算術論理・比較演算部７８ａ～７８ｃの概略構成を示すブロック図である。図６および図２を参照して、算術論理・比較演算部７８ａ～７８ｃの各々は、データバス９０を介してレジスタファイル７６に接続されたＡＬＵ（Arithmetic and Logical Unit）部１３２と、ＡＬＵ部１３２およびデータバス９２を介してレジスタファイル７６に接続され、飽和、最大・最小値検出、絶対値生成処理を行なう飽和処理部１３４と、ＡＬＵ部１３２に接続され、オーバーフロー

の検出とコンディションフラグの生成を行なうフラグ部１３６とを含む。

【００５５】

図７は、バレルシフタ８２ａ～８２ｃの概略構成を示すブロック図である。図７および図２を参照して、バレルシフタ８２ａ～８２ｃの各々は、３２ビットのデータを保持するアキュムレータＭ０およびＭ１を有するアキュムレータ部１４２と、アキュムレータＭ０およびデータバス９０を介してレジスタファイル７６に接続され、アキュムレータＭ０またはレジスタの値を受けるセレクタ１４６と、アキュムレータＭ１およびデータバス９０を介してレジスタファイル７６に接続され、アキュムレータＭ１およびレジスタの値を受けるセレクタ１４８と、セレクタ１４６の出力に接続された上位バレルシフタ１５０と、セレクタ１４８の出力に接続された下位バレルシフタ１５２と、上位バレルシフタ１５０および下位バレルシフタ１５２の出力に接続された飽和処理部１５４とを含む。

【００５６】

飽和処理部１５４の出力は、アキュムレータ部１４２とデータバス９２を介してレジスタファイル７６とに接続されている。

バレルシフタ８２ａ～８２ｃの各々は、構成部品を動作させることにより、データの算術シフト（２の補数体系のシフト）または論理シフト（符号なしシフト）を実行する。通常は、３２ビットもしくは、６４ビットのデータを入出力としている。レジスタファイル７６内のレジスタまたはアキュムレータ部１４２内のアキュムレータに格納された被シフトデータに対して、別のレジスタまたは即値でシフト量が指定される。データは、左６３ビット～右６３ビットの算術または論理シフトが行われ、入力ビット長で出力される。

【００５７】

また、バレルシフタ８２ａ～８２ｃの各々は、ＳＩＭＤ型命令に対して、８、１６、３２、６４ビットのデータをシフトすることができる。例えば、８ビットデータのシフトを４並列で処理することができる。

【００５８】

算術シフトは、２の補数体系のシフトであり、加算や減算時の小数点の位置合わせや、２のべき乗の乗算（２、２の２乗、２の（－１）乗、２の（－２）乗倍

など)等のために行われる。

【００５９】

図８は、除算器８４の概略構成を示すブロック図である。図８および図２を参照して、除算器８４は、３２ビットのデータを保持するアキュムレータＭ０およびＭ１を有するアキュムレータ部１６２と、アキュムレータ部１６２ならびにデータバス９０および９２を介してレジスタファイル７６に接続された除算部１６４とを含む。

【００６０】

除算器８４は、被除数を６４ビット、除数を３２ビットとし、商と剰余を３２ビットずつ出力する。商と剰余を求めるまでに３４サイクルを必要とする。符号付き、符号なし、両方のデータを扱うことが可能である。ただし、被除数と除数において符号の有無の設定は共通とする。その他、オーバーフローフラグ、０除算フラグを出力する機能を有する。

【００６１】

図９は、乗算・積和演算部８０ａおよび８０ｂの概略構成を示すブロック図である。図９および図２を参照して、乗算・積和演算部８０ａおよび８０ｂの各々は、６４ビットのデータを保持するアキュムレータＭ０およびＭ１を有するアキュムレータ部１７２と、各々データバス９０を介してレジスタファイル７６に接続された２入力の３２ビット乗算器（MUL）１７４ａおよび１７４ｂとを含む。

【００６２】

乗算・積和演算部８０ａおよび８０ｂの各々は、さらに、乗算器１７４ａの出力およびアキュムレータ部１７２に接続された６４ビット加算器（Adder）１７６ａと、乗算器１７４ｂの出力およびアキュムレータ部１７２に接続された６４ビット加算器１７６ｂと、６４ビット加算器１７６ａおよび６４ビット加算器１７６ｂの出力に接続された６４ビット加算器１７６ｃと、６４ビット加算器１７６ｂおよび１７６ｃの出力に接続されたセレクタ１７８と、加算器１７６ａの出力、セレクタ１７８の出力、アキュムレータ部１７２およびデータバス９２を介してレジスタファイル７６に接続された飽和処理部（Saturation）１８０とを含む。

【００６３】

乗算・積和演算部８０ａおよび８０ｂの各々は、以下の乗算、積和演算を行う。

・３２×３２ビットのsignedの乗算、積和、積差演算

・３２×３２ビットのunsignedの乗算

・１６×１６ビットの２並列のsignedの乗算、積和、積差演算

・３２×１６ビットの２並列のsignedの乗算、積和、積差演算

これらの演算を整数、固定小数点フォーマットのデータに対して行う。また、これらの演算に対し、丸め、飽和を行う。

【００６４】

図１０は、このようなプロセッサ３０による命令実行時の各パイプライン動作を示すタイミング図である。図２および図１０を参照して、命令フェッチステージでは、命令フェッチ部５２が、ＰＣ部７４に保持されたプログラムカウンタで指定されるアドレスの命令メモリ４０をアクセスし、パケットを命令バッファ５４に転送する。命令割り当てステージでは、命令バッファ５４に保持された命令が命令レジスタ５６ａ～５６ｃに割り当てられる。デコードステージでは、命令レジスタ５６ａ～５６ｃに割り当てられた命令が、命令発行制御部６２からの制御に従い、命令デコーダ６４ａ～６４ｃでそれぞれデコードされる。実行ステージでは、命令デコーダ６４ａ～６４ｃでのデコード結果に基づいて、実行制御部７２が実行部７０の構成部品を動作させ、各種演算を実行する。書き込みステージでは、演算結果をデータメモリ１００またはレジスタファイル７６に格納する。これらの処理により、最高３並列のパイプライン処理が実行可能になる。

【００６５】

図１１は、プロセッサ３０で実行される命令、処理の内容およびそのビットパターンを示す図である。命令“ｌｄ　Ｒｓ，Ｒｄ”は、データメモリ１００の、図１で示したオペレーションのＲｓフィールドで指定されるレジスタ（以下「レジスタＲｓ」という。以下同様である。）の値の番地のデータメモリ１００に記憶されたデータを、レジスタＲｄにロードする処理を示す。そのビットパターンは、図１１で図示したとおりである。

【００６６】

なお、図１１のビットパターンのうち、先頭２ビット（３０および３１ビット目）は、オペレーション群を指定するために用いられ、０ビット目は、並列実行境界情報を指定するために用いられる。上述の先頭２ビットが“０１”のオペレーションは、メモリアクセスに関するものである。先頭２ビットが“１０”のオペレーションは、乗算に関するものである。先頭２ビットが“１１”のオペレーションは、その他の演算に関するものである。

【００６７】

命令“ｓｔ　Ｒｓ，Ｒｄ”は、レジスタＲｓの値を、データメモリ１００のレジスタＲｄで指定される番地にストアする処理を示す。

命令“ｍｕｌ１　Ｒｓ，Ｒｄ”は、レジスタＲｓの値とレジスタＲｄの値との積をレジスタＲｄに書き込む処理を示す。命令“ｍｕｌ２　Ｒｓ１，Ｒｓ２、Ｒｄ”は、レジスタＲｓ１の値とレジスタＲｓ２の値との積をレジスタＲｄに書き込む処理を示す。

【００６８】

命令“ａｄｄ１　Ｒｓ，Ｒｄ”は、レジスタＲｓの値とレジスタＲｄの値との和をレジスタＲｄに書き込む処理を示す。命令“ａｄｄ２　Ｒｓ１，Ｒｓ２，Ｒｄ”は、レジスタＲｓ１の値とレジスタＲｓ２の値との和をレジスタＲｄに書き込む処理を示す。

【００６９】

命令“ｓｕｂ１　Ｒｓ，Ｒｄ”は、レジスタＲｓの値とレジスタＲｄの値との差をレジスタＲｄに書き込む処理を示す。命令“ｓｕｂ２　Ｒｓ１，Ｒｓ２，Ｒｄ”は、レジスタＲｓ１の値とレジスタＲｓ２の値との差をレジスタＲｄに書き込む処理を示す。

【００７０】

命令“ｍｏｖ１　Ｒｓ，Ｒｄ”は、レジスタＲｓの値をレジスタＲｄに書き込む処理を示す。命令“ｍｏｖ２　Ｉｍｍ，Ｒｄ”は、Ｉｍｍフィールドの値をレジスタＲｄに書き込む処理を示す。

【００７１】

命令“ｄｉｖ　Ｒｓ，Ｒｄ”は、レジスタＲｓの値をレジスタＲｄの値で除した商をレジスタＲｄに書き込む処理を示す。命令“ｍｏｄ　Ｒｓ，Ｒｄ”は、レジスタＲｓの値をレジスタＲｄの値で除した剰余をレジスタＲｄに書き込む処理を示す。

【００７２】

[コンパイラ]

次に、上述のプロセッサ３０をターゲットとする本実施の形態に係るコンパイラの一例について図１２～図３８を用いて説明する。

【００７３】

[コンパイラの全体構成]

図１２は、本実施の形態に係るコンパイラ２００の構成を示す機能ブロック図である。このコンパイラ２００は、Ｃ／Ｃ＋＋言語等の高級言語で記述されたソースプログラム２０２を、上述のプロセッサ３０をターゲットプロセッサとする機械語プログラム２０４に変換するクロスコンパイラであり、パーソナルコンピュータ等のコンピュータ上で実行されるプログラムによって実現され、大きく分けて、パーサー部２１０と、中間コード変換部２２０と、最適化部２３０と、コード生成部２４０とから構成される。

【００７４】

パーサー部２１０は、コンパイルの対象となるソースプログラム２０２（インクルードされるヘッダファイルを含む）に対して、予約語（キーワード）等を抽出して字句解析する前置処理部であり、通常のコンパイラが備える解析機能を有する。

【００７５】

中間コード変換部２２０は、パーサー部２１０に接続され、パーサー部２１０から渡されたソースプログラム２０２の各ステートメントを一定規則に基づいて中間コードに変換する処理部である。ここで、中間コードは、典型的には、関数呼び出しの形式で表現されるコード（例えば、「+(int a, int b)」を示すコード；「整数ａに整数ｂを加算する」ことを示す。）である。

【００７６】

最適化部２３０は、中間コード変換部２２０に接続され、中間コード変換部２２０から出力された中間コードについて、命令のオペコードに着目し、命令間の依存関係を崩すことなくプロセッサ３０の消費電力が小さくなるように、命令の配置を行なう命令スケジューリング部２３２と、命令スケジューリング部２３２に接続され、命令スケジューリング部２３２でのスケジュール結果について、命令のレジスタフィールドに着目して、プロセッサ３０の消費電力が小さくなるようにレジスタを割り付けるレジスタ割付部２３４とを含む。

【００７７】

最適化部２３０は、さらに、レジスタ割付部２３４に接続され、レジスタが割り付けられたスケジュール結果について、命令のビットパターンに着目し、命令の依存関係を崩すことなくプロセッサ３０の消費電力が小さくなるように、命令の再配置を行なう命令再スケジューリング部２３６と、命令再スケジューリング部２３６に接続され、命令再スケジューリング部２３６のスケジュール結果について、一定サイクル以上停止しているスロットを検出し、その前後に、当該スロットを停止および復帰させる命令を挿入するスロット停止・復帰命令生成部２３８とを含む。

【００７８】

最適化部２３０は、さらに、スロット停止・復帰命令生成部２３８に接続され、スケジュール結果に基づいて、配置された命令の並列実行境界情報を設定する並列実行境界情報設定部２３９と、命令スケジューリング部２３２、レジスタ割付部２３４および命令再スケジューリング部２３６に接続され、スケジュール結果を各サイクルごとに消費電力が小さくなるように配置しなおすサイクル内配置調整処理部２３７とを含む。

【００７９】

なお、後述する最適化部２３０での処理は、基本ブロック単位に行なわれる。基本ブロックとは、たとえば式や代入文の並びのような、途中から外部への分岐が起こらず、また、外部から途中への分岐も起こらないプログラムの単位を言う。

【００８０】

コード生成部２４０は、最適化部２３０の並列実行境界情報設定部２３９に接続され、並列実行境界情報設定部２３９から出力された中間コードに対して、内部に保持する変換テーブル等を参照することで、全ての中間コードを機械語命令に置き換えることで、機械語プログラム２０４を生成する。

【００８１】

次に、以上のように構成されたコンパイラ２００の特徴的な動作について、具体的な例を示しながら説明する。

【００８２】

[命令スケジューリング部]

図１３は、命令スケジューリング部２３２の動作を示すフローチャートである。命令スケジューリング部２３２の処理では、レジスタのスケジューリングは行なわず、レジスタの個数は無限にあると想定して処理が行なわれる。したがって、以下の説明では、命令スケジューリング部２３２でスケジューリングされるレジスタにはＶｒ（Virtual Register）０、Ｖｒ１など、先頭にＶｒが付されるものとする。

【００８３】

命令スケジューリング部２３２は、中間コード変換部２２０で生成された中間コードに基づいて、命令の依存グラフを作成する（ステップＳ２（以下「ステップ」を省略する。））。依存グラフとは、命令間の依存関係を示したグラフであり、命令ごとにノードを割り付け、依存関係のある命令をエッジで結んだ有向グラフである。依存グラフに関しては、周知の技術である。したがって、その詳細な説明はここでは繰返さない。たとえば、ここでは、図１４（ａ）に示されるような３つの有向グラフからなる依存グラフが作成されるものとする。

【００８４】

命令スケジューリング部２３２は、依存グラフの中から実行可能な命令（ノード）を選択し、そのうち、各スロットのデフォルト論理に合致するように１サイクル目の命令をスケジューリングする（Ｓ４）。例えば、図１４（ａ）の依存グラフでは、ノードＮ１、Ｎ６、Ｎ７、Ｎ１１およびＮ１２の命令のノードがスケジューリング可能であるが、そのうち、ノードＮ１がメモリアクセスに関する命

令であり、ノードＮ１１が乗算命令であり、ノードＮ６がシフト命令であるとする。この場合、ノードＮ１、Ｎ１１およびＮ６がそれぞれ１サイクル目の第１～第３スロットにそれぞれ配置される。配置済みのノードにはフラグが付され、依存グラフは図１４（ｂ）のように更新される。１サイクル目の命令スケジューリング（Ｓ４）の後、図１５に示されるような命令のスケジュール結果が得られる。

【００８５】

命令スケジューリング部２３２は、依存グラフを参照し、配置候補命令集合を生成する（Ｓ８）。すなわち、図１４（ｂ）の例では、ノードＮ２、Ｎ７、Ｎ８およびＮ１２で示される命令が配置候補命令集合となる。

【００８６】

命令スケジューリング部２３２は、配置候補命令集合の中から後述するアルゴリズムに従い、最適な命令を１つ取り出す（Ｓ１２）。

命令スケジューリング部２３２は、取り出された最適命令が実際に配置可能か否かを判断する（Ｓ１４）。配置可能か否かの判断は、最適命令を配置したと仮定した場合の着目サイクルの命令数が、１つ前のサイクルに配置された命令数を超えないか否かにより判断される。これにより、同一数の命令が配置されたサイクルが連続することとなる。

【００８７】

配置可能と判断した場合には（Ｓ１４でＹＥＳ）、その最適命令を仮配置し、配置候補命令集合から削除する（Ｓ１６）。その後、命令スケジューリング部２３２は、さらに命令を配置することが可能か否かを、上述の判断処理（Ｓ１４）と同様にして判断する（Ｓ１８）。配置可能と判断した場合には（Ｓ１８でＹＥＳ）、依存グラフを参照し、新たな配置候補命令が生じた場合には、それを配置候補命令集合に追加する（Ｓ２０）。以上の着目サイクルに対する命令仮配置処理を、配置候補命令がなくなるまで繰返す（Ｓ１０～Ｓ２２）。

【００８８】

なお、最適命令の仮配置処理（Ｓ１６）の後、これ以上、着目サイクルに命令を配置することができないと判断した場合には（Ｓ１８でＮＯ）、命令の仮配置

処理（Ｓ１０～Ｓ２２）のループを抜ける。

【００８９】

命令の仮配置処理（Ｓ１０～Ｓ２２）の後、命令スケジューリング部２３２は、仮配置された命令を確定させ、配置候補命令集合に対するスケジューリングを終了する（Ｓ２４）。その後、配置済みの命令に関しては、依存グラフの対応するノードに配置済みのフラグが付され、依存グラフの更新が行なわれる（Ｓ２６）。

【００９０】

命令スケジューリング部２３２は、一定サイクル以上、同一数の命令が連続配置されているか否かを判断する（Ｓ２７）。一定サイクル以上、同一数の命令が連続配置されていると判断した場合（たとえば、２０サイクル以上２命令が連続配置されている場合や、１０サイクル以上１命令が連続配置されている場合）には（Ｓ２７でＹＥＳ）、命令スケジューリング部２３２は、１サイクルに配置可能な命令の最大数（以下「最大配置可能命令数」という。）を３に設定し（Ｓ２８）、以降のサイクルでは、なるべく、１サイクルに３命令が配置されるようにする。以上の処理を、未配置命令がなくなるまで繰返す（Ｓ６～Ｓ２９）。

【００９１】

図１６は、図１３の最適命令取出し処理（Ｓ１２）の動作を示すフローチャートである。

命令スケジューリング部２３２は、配置候補命令の各々について、着目サイクルの１つ前のサイクルで配置された命令の各々との間で、オペコードのビットパターン間のハミング距離を求める（Ｓ４２）。

【００９２】

たとえば、図１４（ｂ）を参照して、２サイクル目のスケジューリングの開始当初は、ノードＮ２、Ｎ７、Ｎ８およびＮ１２が配置可能である。１サイクル目では、ノードＮ１、Ｎ６およびＮ１１が配置されている。このため、ノードＮ１、Ｎ６およびＮ１１とノードＮ２、Ｎ７、Ｎ８およびＮ１２との間のすべての組み合わせについて、オペコードのビットパターン間のハミング距離が求められることになる。

【００９３】

図１７は、オペコードのビットパターン間のハミング距離の算出方法を説明するための図である。Ｎサイクル目にはすでに、命令“ｌｄ　Ｖｒ１１，Ｖｒ１２”が配置済みであり、Ｎ＋１サイクル目の配置候補命令は、“ｓｔ　Ｖｒ１３，Ｖｒ１４”および“ａｄｄ１　Ｖｒ１３，Ｖｒ１４”であるとする。図１７（ａ）を参照して、オペコード“ｌｄ”および“ｓｔ”は、１２、１６、１７、２４および２５ビット目のビットパターンが異なる。このため、ハミング距離は５である。同様にして、図１７（ｂ）を参照して、オペコード“ｌｄ”および“ａｄｄ１”は、１６、１７、１８、２０、２５、２６、２８および３１ビット目のビットパターンが異なる。このため、ハミング距離は８である。

【００９４】

図１８は、ビット長が異なるオペコード間でのハミング距離の算出方法を説明するための図である。Ｎサイクル目にはすでに、命令“ｌｄ　Ｖｒ１１，Ｖｒ１２”が配置済みであり、Ｎ＋１サイクル目の配置候補命令は、“ｍｕｌ２　Ｖｒ１３，Ｖｒ１４，Ｖｒ１５”および“ｓｔ　Ｖｒ１３，Ｖｒ１４”であるとする。図１８（ａ）を参照して、オペコード“ｌｄ”および“ｍｕｌ２”のように、ビット長が異なるオペコード間では、オペコードの重複部分のビットパターンについてハミング距離が計算される。したがって、オペコードの１６ビット目から３１ビット目までの値に基づいてハミング距離が算出される。オペコード“ｌｄ”および“ｍｕｌ２”は、１６、１８、１９、２２、２３、２５、２６、２７、２８、３０および３１ビット目が異なる。このため、ハミング距離は１１である。図１８（ｂ）を参照して、その他の配置候補命令“ｓｔ　Ｖｒ１３，Ｖｒ１４”についても、図１８（ａ）の例との整合性を確保するため、オペコードの１６ビット目から３１ビット目までの値に基づいてハミング距離が算出される。オペコード“ｌｄ”および“ｓｔ”は、１６、１７、２４および２５ビット目が異なる。このため、ハミング距離は４である。

【００９５】

再度図１６を参照して、命令スケジューリング部２３２は、最小ハミング距離を有する配置候補命令を特定する（Ｓ４３）。図１７および図１８の例では、命

令“ｓｔ　Ｖｒ１３，Ｖｒ１４”がともに配置候補命令として特定される。

【００９６】

命令スケジューリング部２３２は、最小ハミング距離を有する配置候補命令が２以上あるか否かを判断する（Ｓ４４）。最小ハミング距離を有する配置候補命令が１つの場合には（Ｓ４４でＮＯ）、その命令を最適命令とする（Ｓ５６）。

【００９７】

最小ハミング距離を有する配置候補命令が２つ以上ある場合には、（Ｓ４４でＹＥＳ）、それらの配置候補命令のうち命令が配置されていない空きスロットのデフォルト論理に合致するものがあるか否かを判断する（Ｓ４６）。

【００９８】

デフォルト論理に合致する配置候補命令がなければ（Ｓ４６でＮＯ）、最小ハミング距離を有する２以上の配置候補命令のいずれかを任意に選択し、最適命令とする（Ｓ５４）。

【００９９】

デフォルト論理に合致する配置候補命令があり、かつその個数が１つであれば（Ｓ４６でＹＥＳ、Ｓ４８でＮＯ）、デフォルト論理に合致する配置候補命令を最適命令とする（Ｓ５２）。

【０１００】

デフォルト論理に合致する配置候補命令があり、かつその個数が２つ以上であれば（Ｓ４６でＹＥＳ、Ｓ４８でＹＥＳ）、デフォルト論理に合致する２以上の配置可能命令のうちのいずれかを任意に選択して最適命令とする（Ｓ５０）。

【０１０１】

[サイクル内配置調整処理部]

図１９は、サイクル内配置調整処理部２３７の動作を示すフローチャートである。サイクル内配置調整処理部２３７は、命令スケジューリング部２３２でのスケジュール結果に基づいて、各サイクル内での命令の配置の調整を行なう。

【０１０２】

サイクル内配置調整処理部２３７は、スケジュール結果の２サイクル目から最終サイクルまでのうち、着目しているサイクルの３つの命令について並べ替えを

行ない、６通りの命令並びを作成する（Ｓ６１）。図２０は、このようにして作成された６通りの命令並びの一例を示す図である。

【０１０３】

サイクル内配置調整処理部２３７は、後述する６通りの命令並びの各々についてハミング距離の和を求める処理（Ｓ６２～Ｓ６７）を実行する。６通りの命令並びの各々について求められたハミング距離の和のうち最小のハミング距離の和をとる命令並びを選択し、その命令並びの並びになるように命令の並べ替えを行なう（Ｓ６８）。以上の処理を、２サイクル目から最終サイクルまで繰返す（Ｓ６０～Ｓ６９）。

【０１０４】

次に、６通りの命令並びの各々についてハミング距離の和を求める処理（Ｓ６２～Ｓ６７）について説明する。サイクル内配置調整処理部２３７は、各命令並びの各スロットについて、着目命令と１つ前のサイクルの命令とのオペコードのビットパターン間のハミング距離を求める（Ｓ６４）。ハミング距離を求める処理（Ｓ６４）を３つのスロットの命令のすべてについて行ない（Ｓ６３～Ｓ６５）、３つのスロットの命令の各々についてハミング距離の和を求める（Ｓ６６）。以上の処理を、６通りの命令並びのすべてについて行なう（Ｓ６２～Ｓ６７）。

【０１０５】

図２１は、配置された命令の一例を示す図である。Ｎサイクル目には、第１スロット、第２スロットおよび第３スロットで実行される命令として、“ｌｄ　Ｖｒ１０，Ｖｒ１１”、“ｓｕｂｌ　Ｖｒ１２，Ｖｒ１３”および“ａｄｄｌ　Ｖｒ１４，Ｖｒ１５”がそれぞれ配置されているものとする。Ｎ＋１サイクル目には、第１スロット、第２スロットおよび第３スロットで実行される命令として、“ｓｔ　Ｖｒ１６，Ｖｒ１７”、“ｍｕｌ　Ｖｒ１８，Ｖｒ１９”および“ｍｏｄ　Ｖｒ２０，Ｖｒ２１”がそれぞれ配置されているものとする。

【０１０６】

図２２は、命令並び作成処理（Ｓ６１）を説明するための図である。たとえば、図２１に示すＮ＋１サイクル目に配置された３つの命令より、図２２（ａ）～

（ｆ）に示す６つの命令並びが作成される。

【０１０７】

図２３は、オペコードのハミング距離算出処理（Ｓ６４）を説明するための図である。たとえば、図２１に示すＮサイクル目の命令並びと、図２２（ｃ）に示すＮ＋１サイクル目の命令並びとの間で、スロットごとにオペコードのハミング距離を算出すると、第１スロット、第２スロットおよび第３スロットにおけるハミング距離は、それぞれ１０、９および５となる。

【０１０８】

したがって、図２３の例におけるハミング距離の和は２４となる。ハミング距離和算出処理（Ｓ６６）では、このようにして、図２１に示すＮサイクル目の命令並びと、図２２（ａ）～（ｆ）に示す６通りの命令並びの各々との間でハミング距離の和が求められ、それぞれ、１４、１６、２４、２２、２４および２０となる。命令並び選択処理（Ｓ６８）では、６通りの命令並びのうち、最小のハミング距離の和をとる図２２（ａ）の命令並びが選択される。

【０１０９】

[レジスタ割付部]

図２４は、レジスタ割付部２３４の動作を示すフローチャートである。レジスタ割付部２３４では、命令スケジューリング部２３２およびサイクル内配置調整処理部２３７でのスケジュール結果に基づいて、実際にレジスタの割付を行なっていく。

【０１１０】

レジスタ割付部２３４は、ソースプログラム２０２から割付対象（変数）を抜き出し、各割付対象の生存区間およびその優先度を求める（Ｓ７２）。生存区間とは、プログラム中で、変数が定義されてから、その参照が終了するまでの区間を言う。したがって、同一の変数であっても、複数の生存区間が存在する場合がある。優先度とは、割付対象の生存区間長およびその参照頻度で決定される。その詳細な説明は、本発明の本質的事項ではないため、省略する。

【０１１１】

レジスタ割付部２３４は、割付対象より干渉グラフを作成する（Ｓ７４）。干

渉グラフとは、同一のレジスタを割り付けることができない割付対象の条件を示したグラフである。次に、干渉グラフの作成方法について説明する。

【０１１２】

図２５は、割付対象となる変数の生存区間を示す図である。ここでは、変数Ｉ、ＪおよびＫの３つの変数を割付対象とした例を示す。

変数Ｉは、ステップＴ１で定義されて、ステップＴ５で最終参照される。また、変数Ｉは、ステップＴ８で定義されて、ステップＴ１０で最終参照される。したがって、変数Ｉは、２つの生存区間を有することとなる。先の生存区間での変数Ｉを変数Ｉ１と定義し、後の生存区間での変数を変数Ｉ２と定義することとする。変数Ｊは、ステップＴ２で定義されて、ステップＴ４で最終参照される。

【０１１３】

変数Ｋは、ステップＴ３で定義されて、ステップＴ６で最終参照される。また、変数Ｋは、ステップＴ７で定義されてステップＴ９で最終参照される。したがって、変数Ｉと同様、変数Ｋは２つの生存区間を有することとなる。先の生存区間での変数Ｋを変数Ｋ１と定義し、後の生存区間での変数Ｋを変数Ｋ２と定義する。

【０１１４】

変数Ｉ１、Ｉ２、Ｊ、Ｋ１およびＫ２には、以下に示すような生存区間の重なりが生じる。すなわち、変数Ｉ１およびＪの生存区間は、ステップＴ２～Ｔ４で重なりを有する。変数ＪおよびＫ１の生存区間は、ステップＴ３～Ｔ４で重なりを有する。変数Ｉ１およびＫ１の生存区間は、ステップＴ３～Ｔ５で重なりを有する。変数Ｉ２およびＫ２は、ステップＴ８～Ｔ９で重なりを有する。このように、生存区間が重なる変数同士は、同一のレジスタに割り付けることはできない。このため、割付対象となる変数をノードとし、生存区間が重なる変数同士をエッジで結んだものが干渉グラフとなる。

【０１１５】

図２６は、図２５の例に基づいて作成された変数の干渉グラフを示す図である。ノードＩ１、Ｋ１およびＪは相互にエッジにより接続されている。このため、変数Ｉ１、Ｋ１およびＪの間には相互に生存区間が重なる区間があり、これら３

つの変数に同じレジスタを割り付けることはできないことがわかる。同様に、ノードＩ２およびＫ２はエッジにより接続されている。このため、変数Ｉ２およびＫ２に同じレジスタを割り付けることはできないことがわかる。

【０１１６】

しかし、エッジにより接続されていないノード間には依存関係が存在しない。たとえば、ノードＪおよびＫ２はエッジにより接続されていない。このため、変数ＪおよびＫ２には生存区間の重なりがなく、同じレジスタを割り付けることができることがわかる。

【０１１７】

再度図２４を参照して、レジスタ割付部２３４は、レジスタ割付を行なっていない割付対象のうち、優先度が最も高い割付対象を選択する（Ｓ８０）。命令スケジューリング部２３２は、割付対象を割り付けるレジスタとして、同一スロットで割付対象を参照する命令の直前に実行される命令のうち、同一フィールドのレジスタ番号と同一番号のレジスタが割り付け可能か否かを判断する（Ｓ８２）。割り付け可能か否かの判断は、上述した干渉グラフを参照することにより行なわれる。

【０１１８】

図２７は、命令スケジューリングの途中結果を示す図である。たとえば、図２７（ａ）を参照して、現在の割付対象は、第１スロットの（Ｎ＋１）番目のサイクルのソースオペランド（レジスタＶｒ５）に割り付けられるものとする。レジスタＶｒ５は、上述したように仮に設けられたレジスタである。このため、図２４のレジスタ割付可能判断処理（Ｓ８２）では、割付対象として、Ｎ番目のサイクルの同一フィールドで使用されるレジスタ（ここでは、レジスタＲ０）が割付可能か否かを判断することになる。図２７（ｂ）は、Ｖｒ５にレジスタＲ０を割付けた場合の命令のビットパターンを示している。このように、連続するサイクル間で同一のレジスタをアクセスすると、レジスタの特性により消費電力を削減することができる。

【０１１９】

同一番号のレジスタが割り付け可能と判断された場合には（Ｓ８２でＹＥＳ）

、レジスタ割付部２３４は、割付対象に、上述の同一番号のレジスタを割り付ける（Ｓ８４）。同一番号のレジスタを割り付けることができないと判断された場合には（Ｓ８２でＮＯ）、レジスタ割付部２３４は、割付可能なレジスタのレジスタ番号（２進表現）の中で、先行サイクルの同一スロットの同一フィールドのレジスタ番号との間のハミング距離が最小となるものを求める（Ｓ８６）。図２７（ｃ）は、レジスタＲ０のレジスタ番号（０００００）とのハミング距離が最小となるレジスタ番号（００００１）を有するレジスタＲ１が、使用可能なレジスタの中から選択された例を示している。

【０１２０】

ハミング距離が最小となる割付可能なレジスタが１つしかない場合には（Ｓ８８でＮＯ）、割付対象に当該レジスタを割り付ける（Ｓ９２）。ハミング距離が最小となる割付可能なレジスタが２つ以上ある場合には（Ｓ８８でＹＥＳ）、２つ以上の割付可能なレジスタのいずれかを任意に選択し、割付対象に割り付ける（Ｓ９０）。以上の処理を、割付対象がなくなるまで行なう（Ｓ７８～Ｓ９４）。

【０１２１】

レジスタ割付部２３４での処理の後、サイクル内配置調整処理部２３７は、レジスタ割付部２３４でのスケジュール結果に基づいて、各サイクル内での命令の配置の調整を行なう。サイクル内配置調整処理部２３７で実行される処理は、図１９および図２０を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０１２２】

[命令再スケジューリング部]

図２８は、命令再スケジューリング部２３６の動作を示すフローチャートである。命令再スケジューリング部２３６は、命令スケジューリング部２３２、レジスタ割付部２３４およびサイクル内配置調整処理部２３７で実行された処理により、プロセッサ３０で動作可能にスケジューリングされた命令のスケジュール結果を、再度スケジューリングしなおす処理を行なう。すなわち、命令再スケジューリング部２３６は、レジスタ割付部２３４にて実レジスタが確定した命令列に

対して、再度スケジューリングを行なうものである。

【０１２３】

命令再スケジューリング部２３６は、スケジュール結果の中から冗長な命令を削除する（Ｓ１１２）。たとえば、命令“ｍｏｖ１　Ｒ０，Ｒ０”は、レジスタＲ０の内容をレジスタＲ０に書き込む処理であるため、冗長な命令である。また、同一サイクルの第１スロットの命令が“ｍｏｖ２　４，Ｒ１”であり、第２スロットの命令が“ｍｏｖ２　５，Ｒ１”である場合には、それぞれ４および５をレジスタＲ１に書き込む命令である。本実施の形態では、番号の大きいほうのスロットの命令が優先的に実行されることとする。このため、第１スロットの命令“ｍｏｖ２　４，Ｒ１”は、冗長な命令である。

【０１２４】

冗長な命令を削除すると、命令の依存関係が変化する場合がある。このため、命令再スケジューリング部２３６は、依存グラフの再構築を行なう（Ｓ１１４）。命令再スケジューリング部２３６は、依存グラフの中から実行可能な命令（ノード）を選択し、そのうち、各スロットのデフォルト論理に合致するように１サイクル目の命令をスケジューリングする（Ｓ１１５）。１サイクル目の命令に対応する依存グラフのノードには、配置済みのフラグが付される。

【０１２５】

命令再スケジューリング部２３６は、依存グラフを参照し、配置候補命令集合を生成する（Ｓ１１８）。命令再スケジューリング部２３６は、配置候補命令集合の中から後述するアルゴリズムに従い、最適な命令を１つ取り出す（Ｓ１２２）。

【０１２６】

命令再スケジューリング部２３６は、取り出された最適命令が実際に配置可能か否かを判断する（Ｓ１２４）。配置可能か否かの判断は、図１３のＳ１４の判断と同様である。このため、その詳細な説明はここでは繰返さない。

【０１２７】

配置可能と判断した場合には（Ｓ１２４でＹＥＳ）、その最適命令を仮配置し、配置候補命令集合から削除する（Ｓ１２６）。その後、命令再スケジューリン

グ部２３６は、さらに命令を配置することが可能か否かを、上述の配置可能判断（Ｓ１２４）と同様にして判断する（Ｓ１２８）。配置可能と判断した場合には（Ｓ１２８でＹＥＳ）、依存グラフを参照し、新たな配置候補命令が生じた場合には、それを配置候補命令集合に追加する（Ｓ１３０）。以上の処理を、配置候補命令がなくなるまで繰返す（Ｓ１２０～Ｓ１３２）。

【０１２８】

なお、最適命令の仮配置処理（Ｓ１２６）の後、これ以上、着目サイクルに命令を配置することができないと判断した場合には（Ｓ１２８でＮＯ）、最適命令の仮配置処理（Ｓ１２０～Ｓ１３２）のループを抜ける。

【０１２９】

最適命令の仮配置処理（Ｓ１２０～Ｓ１３２）の後、命令再スケジューリング部２３６は、仮配置された命令を確定させ、配置候補命令集合に対するスケジューリングを終了する（Ｓ１３４）。その後、配置済みの命令に関しては、依存グラフの対応するノードに配置済みのフラグが付され、依存グラフの更新が行なわれる（Ｓ１３６）。

【０１３０】

命令再スケジューリング部２３６は、一定サイクル以上、同一数の命令が連続配置されているか否かを判断する（Ｓ１３７）。一定サイクル以上、同一数の命令が連続配置されていると判断した場合には（Ｓ１３７でＹＥＳ）、命令再スケジューリング部２３６は、最大配置可能命令数を３に設定し（Ｓ１３８）、以降のサイクルでは、なるべく、１サイクルに３命令が配置されるようにする。以上の処理を、未配置命令がなくなるまで繰返す（Ｓ１１６～Ｓ１３９）

図２９は、図２８の最適命令取出し処理（Ｓ１２２）の動作を示すフローチャートである。命令再スケジューリング部２３６は、配置候補命令のうち、着目サイクルの１つ前のサイクルの同一スロットで実行される命令と比較して、同一のレジスタ番号を有するフィールドの個数を求め、当該個数が最大の配置候補命令を特定する（Ｓ１５２）。

【０１３１】

図３０は、配置候補命令特定処理（Ｓ１５２）を説明するための図である。Ｎ

サイクル目の第１スロットで実行される命令として“ａｄｄ１　Ｒ０，Ｒ２”が配置されているものとし、Ｎ＋１サイクル目の第１スロットに配置可能な命令として、図３０（ａ）に示す“ｓｕｂ１　Ｒ０，Ｒ１”と、図３０（ｂ）に示す“ｄｉｖ　Ｒ０，Ｒ２”とがあるものとする。図３０（ａ）に示すように、当該配置位置に命令“ｓｕｂ１　Ｒ０，Ｒ１”を配置した場合には、同一のレジスタ番号を有するフィールドは、レジスタＲ０（レジスタ番号００００００）が配置されたフィールドのみである。このため、同一のレジスタ番号を有するフィールドの個数は１つである。図３０（ｂ）に示すように、当該配置位置に命令“ｄｉｖ　Ｒ０，Ｒ２”を配置した場合には、レジスタＲ０（レジスタ番号００００００）およびレジスタＲ２（０００１０）がそれぞれ配置された２つのフィールドが同一のレジスタ番号を有する。このため、同一のレジスタ番号を有するフィールドの個数は２つである。

【０１３２】

当該個数が最大の配置候補命令が１つしかない場合には（Ｓ１５４でＮＯ）、その配置候補命令を最適命令とする（Ｓ１７４）。

当該個数が最大の配置候補命令がないか、または２つ以上ある場合には（Ｓ１５４でＹＥＳ）、命令再スケジューリング部２３６は、配置候補命令の各々について、１つ前のサイクルの同一スロットで実行される命令と比較して、命令のビットパターンのハミング距離が最小のものを求める（Ｓ１５６）。

【０１３３】

図３１は、配置候補命令特定処理（Ｓ１５６）を説明するための図である。Ｎサイクル目の第１スロットで実行される命令として“ｍｕｌ１　Ｒ３，Ｒ１０”が配置されているものとし、Ｎ＋１サイクル目の第１スロットに配置可能な命令として、図３１（ａ）に示す“ａｄｄ１　Ｒ２，Ｒ４”と、図３０（ｂ）に示す“ｓｕｂ２　Ｒ１１，Ｒ０，Ｒ２”とがあるものとする。それぞれの命令のビットパターンは図示するとおりである。図３１（ａ）に示すように、当該配置位置に命令“ａｄｄ１　Ｒ２，Ｒ４”を配置した場合には、命令“ｍｕｌ１　Ｒ３，Ｒ１０”とのハミング距離は１０である。図３１（ｂ）に示すように、当該配置位置に命令“ｓｕｂ２　Ｒ１１，Ｒ０，Ｒ２”を配置した場合には、命令“ｍｕ

ｌｌ　Ｒ３，Ｒ１０”とのハミング距離は８である。このため、配置候補命令として“ｓｕｂ２　Ｒ１１，Ｒ０，Ｒ２”が特定される。

【０１３４】

最小ハミング距離を有する配置候補命令が１つの場合には（Ｓ１５８でＮＯ），当該配置候補命令を最適命令とする（Ｓ１７２）。

最小ハミング距離を有する配置候補命令が２つ以上ある場合には（Ｓ１５８でＹＥＳ）、２つ以上の配置候補命令のうち、当該配置候補命令が実行されるスロットのデフォルト論理に合致する配置候補命令を特定する（Ｓ１６０）。

【０１３５】

図３２は、配置候補命令特定処理（Ｓ１６０）を説明するための図である。Ｎサイクル目の第１スロットで実行される命令として“ｓｔ　Ｒ１，Ｒ１３”が配置されているものとし、Ｎ＋１サイクル目の第１スロットに配置可能な命令として、図３２（ａ）に示す“ｌｄ　Ｒ３０，Ｒ１８”と、図３２（ｂ）に示す“ｓｕｂ１　Ｒ８，Ｒ２”とがあるものとする。それぞれの命令のビットパターンは図示するとおりである。第１スロットのデフォルト論理は、上述したようにメモリアクセスに関する命令である。これは、命令の先頭２ビットが“０１”であることより判別可能である。命令“ｌｄ　Ｒ３０，Ｒ１８”の先頭２ビットは“０１”であるため、第１スロットのデフォルト論理に合致するが、命令“ｓｕｂ１　Ｒ８，Ｒ２”の先頭２ビットは“１１”であるため、第１スロットのデフォルト論理には合致しない。このため、配置候補命令として“ｌｄ　Ｒ３０，Ｒ１８”が特定される。

【０１３６】

デフォルト論理に合致する配置候補命令がなければ（Ｓ１６２でＮＯ）、最小ハミング距離を有する配置候補命令のうちのいずれかを任意に選択し、最適命令とする（Ｓ１７０）。

【０１３７】

デフォルト論理に合致する配置候補命令があり、かつその個数が１つであれば（Ｓ１６２でＹＥＳ、Ｓ１６４でＮＯ）、デフォルト論理に合致する配置候補命令を最適命令とする（Ｓ１６８）。

【０１３８】

デフォルト論理に合致する配置候補命令があり、かつその個数が２つ以上であれば（Ｓ１６２でＹＥＳ、Ｓ１６４でＹＥＳ）、デフォルト論理に合致する配置候補命令のうちのいずれかを任意に選択し、最適命令とする（Ｓ１６６）。

【０１３９】

命令再スケジューリング部２３６での処理の後、サイクル内配置調整処理部２３７は、命令再スケジューリング部２３６でのスケジュール結果に基づいて、各サイクル内での命令の配置の調整を行なう。サイクル内配置調整処理部２３７で実行される処理は、図１９および図２０を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０１４０】

以上、命令再スケジューリング部２３６の動作について説明を行なったが、コンパイル時のオプションまたはソースプログラム中に記述されたプラグマに従って、１つのサイクルで使用するスロットの個数の制限を行なってもよい。プラグマとは、プログラムの意味を変更することなくコンパイラへの最適化の指針を与える記述のことを言う。たとえば、例１に示すように、Ｃ言語で記述されたソースプログラムをコンパイルする際のオプションとして“－ｐａｒａ”を設け、その後に続く数字でスロットの数を規定する。例１では、ソースプログラム“ｆｏｏ．ｃ”がＣコンパイラによりコンパイルされるが、スケジュール結果の各サイクルには、必ず２命令が配置されることとなる。

【０１４１】

また、例２に示すように、ソースプログラム中に記述された各関数について、使用されるスロットの個数をプラグマで定義してもよい。例２では、関数ｆｕｎｃを実行する際に使用されるスロットの個数が１つと規定されている。このため、スケジュール結果のうち、関数ｆｕｎｃを実行するサイクルの各々には、必ず１命令のみが配置されることとなる。

（例１）

ｃｃ　－ｐａｒａ　２　ｆｏｏ．ｃ

（例２）

```
#pragma  para=1  func
  int  func (void) {
       ......
  }
```

なお、オプションとプラグマとが同時に設定された場合には、値の小さいほうが優先されるようにしてもよい。たとえば、例１に示すソースプログラム“ｆｏｏ．ｃ”中に、例２に示す関数ｆｕｎｃおよびそのプラグマが指定されている場合には、原則として、２スロットの並列処理が実行されるが、関数ｆｕｎｃを実行するサイクルでは１スロットのみで処理が実行されるように、スケジュール結果が作成される。

【０１４２】

また、オプションおよびプラグマに関しては、命令再スケジューリング部２３６のみならず、命令スケジューリング部２３２またはレジスタ割付部２３４での動作で考慮されるようにしてもよい。

【０１４３】

[スロット停止・復帰命令生成部]

図３３は、スロット停止・復帰命令生成部２３８の動作を示すフローチャートである。スロット停止・復帰命令生成部２３８は、命令再スケジューリング部２３６でのスケジュール結果から一定サイクル（たとえば4サイクル）以上、特定の１つのスロットのみが連続使用されている区間を検出する（Ｓ１８２）。スロット停止・復帰命令生成部２３８は、上記区間の１サイクル前の空きスロット位置に残りの２つのスロットを停止させる命令を挿入する（Ｓ１８４）。１サイクル前に命令を挿入する空きスロット位置がない場合には、１サイクル追加し、上記命令を挿入する。

【０１４４】

次に、スロット停止・復帰命令生成部２３８は、上記区間の１サイクル後の空きスロット位置に停止させておいた２つのスロットを復帰させる命令を挿入する（Ｓ１８６）。サイクル後に命令を挿入する空きスロット位置がない場合には、１サイクル追加し、上記命令を追加する。

【０１４５】

図３４は、命令が配置されたスケジュール結果の一例を示す図である。１０サイクル目から１８サイクル目までの９サイクルは第１スロットのみが連続使用されている。このため、９サイクル目の空きスロットに、第１スロットのみを動作させ残りの２つのスロットを停止させる命令（“ｓｅｔ１　１”）が書き込まれる。また、１９サイクル目の空きスロットに、残りの２つのスロットを復帰させる命令（“ｃｌｅａｒ１　１”）が書き込まれる。図３５は、図３３の特定の１スロットのみが連続使用されている場合の処理（Ｓ１８２～Ｓ１８６）で命令が書き込まれたスケジュール結果の一例を示す図である。

【０１４６】

再度図３３を参照して、スロット停止・復帰命令生成部２３８は、スケジュール結果から一定サイクル（たとえば４サイクル）以上、特定の２つのスロットのみが連続使用されている区間を検出する（Ｓ１８８）。スロット停止・復帰命令生成部２３８は、上記区間の１サイクル前の空きスロット位置に残りの１つのスロットを停止させる命令を挿入する（Ｓ１９０）。１サイクル前に命令を挿入する空きスロット位置がない場合には、１サイクル追加し、上記命令を挿入する。

【０１４７】

次に、スロット停止・復帰命令生成部２３８は、上記区間の１サイクル後の空きスロット位置に停止させておいた１つのスロットを復帰させる命令を挿入する（Ｓ１９２）。サイクル後に命令を挿入する空きスロット位置がない場合には、１サイクル追加し、上記命令を追加する。

【０１４８】

図３５のスケジュール結果では、４サイクル目から８サイクル目までの５サイクルは第１および第２スロットのみが使用され、第３スロットは使用されていない。このため、その前後のサイクルに第３スロットを停止させる命令（“ｓｅｔ２　１２”）および復帰させる命令（“ｃｌｅａｒ２　１２”）をそれぞれ書き込む必要がある。しかし、３サイクル目および９サイクル目の双方ともにすべてのスロットに命令が配置されている。このため、スロット停止・復帰命令生成部２３８は、４サイクル目の前および８サイクル目の後に１サイクルずつ新たなサ

イクルを挿入し、上記２命令をそれぞれのサイクルに書き込む。図３６は、図３３の特定の２スロットのみが連続使用されている場合の処理（Ｓ１８８～Ｓ１９２）で命令が書き込まれたスケジュール結果の一例を示す図である。

【０１４９】

なお、本実施の形態では、命令は、第１スロット、第２スロット、第３スロットの順に配置されることを前提としている。このため、２つのスロットが動作している場合には、必ず第３スロットが動作しておらず、１つのスロットのみが動作している場合には、必ず第２スロットと第３スロットとが動作していないことになる。

【０１５０】

また、プロセッサ３０には、３２ビットのプログラム状態レジスタ（図示せず）が設けられている。図３７は、プログラム状態レジスタの一例を示す図である。たとえば、１５および１６ビットの２ビットで動作しているスロットの数を表すことができる。この場合、図３７（ａ）～（ｄ）は、動作しているスロットの数がそれぞれ０～３であることを示している。

【０１５１】

図３８はプログラム状態レジスタの他の一例を示す図である。このプログラム状態レジスタでは、１４ビット目が第１スロットに、１５ビット目が第２スロットに、１６ビット目が第３スロットに対応している。各ビットの値が「１」であれば、そのスロットが動作していることを示し、「０」であれば、そのスロットが停止していることを示す。たとえば、図３８（ｂ）のプログラム状態レジスタでは、第１スロットが停止しており、第２および第３スロットが動作していることを示している。

【０１５２】

上述した命令“ｓｅｔ１”または“ｓｅｔ２”でプログラム状態レジスタに保持された値が書き換えられる。

以上、本実施の形態におけるコンパイラについて説明したが、コンパイラ２００の各部は以下のように変形可能である。次に、その変形例について順次説明を行なう。

【０１５３】

［コンパイラの各部の変形例］

［命令再スケジューリング部２３６の動作の変形例］

本実施の形態では、図２８および図２９を参照して、命令再スケジューリング部２３６の動作について説明したが、図２９を参照して説明した図２８の最適命令取出し処理（Ｓ１２２）の代わりに、図３９に示す最適命令取出し処理を行なってもよい。

【０１５４】

図３９は、図２８の最適命令取出し処理（Ｓ１２２）の他の動作を示すフローチャートである。

命令再スケジューリング部２３６は、図２９の最小ハミング距離を求める処理（Ｓ１５６）の代わりに、以下に示す方法で最小ハミング距離を求める。すなわち、命令再スケジューリング部２３６は、配置候補命令のうち、１つ前のサイクルの同一スロットで実行される命令と比較して、レジスタフィールドのビットパターンのハミング距離が最小のものを求める（Ｓ２１２）。

【０１５５】

図４０は、配置候補命令特定処理（Ｓ２１２）を説明するための図である。Ｎサイクル目の第１スロットで実行される命令として“ａｄｄ１　Ｒ０，Ｒ２”が配置されているものとし、Ｎ＋１サイクル目の第１スロットに配置可能な命令として図４０（ａ）に示す“ｓｕｂ１　Ｒ３，Ｒ１”と、図４０（ｂ）に示す“ｄｉｖ　Ｒ７，Ｒ１”とがあるものとする。それぞれの命令のビットパターンは図示するとおりである。図４０（ａ）に示すように、当該配置位置に命令“ｓｕｂ１　Ｒ３，Ｒ１”を配置した場合には、命令“ａｄｄ１　Ｒ０，Ｒ２”とのレジスタフィールド間のハミング距離は４である。図４０（ｂ）に示すように、当該配置位置に命令“ｄｉｖ　Ｒ７，Ｒ１”を配置した場合には、命令“ａｄｄ１　Ｒ０，Ｒ２”とのレジスタフィールド間のハミング距離は５である。このため、配置候補命令として、“ａｄｄ１　Ｒ０，Ｒ２”が特定される。

【０１５６】

その他の処理（Ｓ１５２～Ｓ１５４、Ｓ１５８～Ｓ１７４）は、図２９で説明

したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０１５７】

[サイクル内配置調整処理部２３７の第１変形例]

サイクル内配置調整処理部２３７は、図１９を参照して説明した処理の代わりに、図４１に示す処理を実行してもよい。

【０１５８】

図４１は、サイクル内配置調整処理部２３７の動作の第１変形例を示すフローチャートである。

サイクル内配置調整処理部２３７は、図１９のハミング距離を求める処理（Ｓ６４）の代わりに、以下に示す方法で最小ハミング距離を求める。すなわち、サイクル内配置調整処理部２３７は、各命令並びの各スロットについて、着目命令と１つ前のサイクルの命令との間で、ビットパターン間のハミング距離を求める（Ｓ２２２）。その他の処理（Ｓ６０～Ｓ６３、Ｓ６５～Ｓ６９）は、図１９を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０１５９】

図４２は、命令のハミング距離算出処理（Ｓ２２２）を説明するための図である。たとえば、図２１に示すＮサイクル目の命令並びと、図２２（ｃ）に示すＮ＋１サイクル目の命令並びとの間で、スロットごとに命令のハミング距離を算出すると、第１スロット、第２スロットおよび第３スロットにおけるハミング距離は、それぞれ１２、１１および１１となる。

【０１６０】

したがって、図４２の例におけるハミング距離の和は３４である。ハミング距離和算出処理（Ｓ６６）では、このようにして、図２１に示すＮサイクル目の命令並びと、図２２（ａ）～（ｆ）に示す６通りの命令並びの各々との間でハミング距離の和が求められ、それぞれ２８、２６、３４、２８、３４および３０となる。命令並び選択処理（Ｓ６８）では、６通りの命令並びのうち、最小のハミング距離の和をとる図２２（ｂ）の命令並びが選択される。

【０１６１】

なお、本変形例のハミング距離を求める処理（Ｓ２２２）では、レジスタが割付けられていることが前提となっている。このため、本変形例のサイクル内配置調整処理部２３７での処理は、レジスタが割付けられていない命令スケジューリング部２３２での処理の後に実行することはできず、レジスタ割付部２３４での処理の後または命令再スケジューリング部２３６での処理の後に実行される。

【０１６２】

［サイクル内配置調整処理部２３７の第２変形例］

サイクル内配置調整処理部２３７は、図１９を参照して説明した処理の代わりに、図４３に示す処理を実行してもよい。

【０１６３】

図４３は、サイクル内配置調整処理部２３７の動作の第２変形例を示すフローチャートである。

サイクル内配置調整処理部２３７は、図１９のハミング距離を求める処理（Ｓ６４）の代わりに、以下に示す方法で最小ハミング距離を求める。すなわち、サイクル内配置調整処理部２３７は、各命令並びの各スロットについて、着目命令と１つ前のサイクルの命令とのレジスタフィールドのビットパターン間のハミング距離を求める（Ｓ２３２）。その他の処理（Ｓ６０～Ｓ６３、Ｓ６５～Ｓ６９）は、図１９を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０１６４】

図４４は、レジスタフィールドのハミング距離算出処理（Ｓ２３２）を説明するための図である。たとえば、図２１に示すＮサイクル目の命令並びと、図２２（ｃ）に示すＮ＋１サイクル目の命令並びとの間で、スロットごとにレジスタフィールドのハミング距離を算出すると、第１スロット、第２スロットおよび第３スロットにおけるハミング距離は、それぞれ２、２および６となる。

【０１６５】

したがって、図４４の例におけるハミング距離の和は１０となる。ハミング距離和算出処理（Ｓ６６）では、このようにして、図２１に示すＮサイクル目の命令並びと、図２２（ａ）～（ｆ）に示す６通りの命令並びの各々との間でハミン

グ距離の和が求められ、それぞれ１４、１０、１０、６、１０および１０となる。命令並び選択処理（Ｓ６８）では、６通りの命令並びのうち、最小のハミング距離の和をとる図２２（ｄ）の命令並びが選択される。

【０１６６】

なお、本変形例のハミング距離を求める処理（Ｓ２３２）では、レジスタが割付けられていることが前提となっている。このため、本変形例のサイクル内配置調整処理部２３７での処理は、レジスタが割付けられていない命令スケジューリング部２３２での処理の後に実行することはできず、レジスタ割付部２３４での処理の後または命令再スケジューリング部２３６での処理の後に実行される。

【０１６７】

[サイクル内配置調整処理部２３７の第３変形例]

サイクル内配置調整処理部２３７は、図１９を参照して説明した処理の代わりに、図４５に示す処理を実行してもよい。

【０１６８】

図４５は、サイクル内配置調整処理部２３７の動作の第３変形例を示すフローチャートである。

サイクル内配置調整処理部２３７は、図１９のハミング距離を求める処理（Ｓ６４）の代わりに、以下の処理を実行する。すなわち、サイクル内配置調整処理部２３７は、各命令並びの各スロットについて、着目命令と１つ前のサイクルの命令との間で、同一のレジスタ番号を有するレジスタフィールドの個数を求める（Ｓ２４２）。

【０１６９】

また、サイクル内配置調整処理部２３７は、図１９のハミング距離の和を求める処理（Ｓ６６）の代わりに、以下の処理を実行する。すなわち、サイクル内配置調整処理部２３７は、３つのスロットの命令の各々について求められた同一のレジスタ番号を有するレジスタフィールドの個数の和を求める（Ｓ２４４）。

【０１７０】

さらに、サイクル内配置調整処理部２３７は、図１９の命令の並べ替え処理（Ｓ６８）の代わりに、以下の処理を実行する。すなわち、サイクル内配置調整処

理部２３７は、６通りの命令並びの各々について求められたレジスタフィールドの個数の和のうち、最大のレジスタフィールドの個数の和をとる命令並びを選択し、その命令並びの並びになるように命令の並べ替えを行なう（Ｓ２４６）。その他の処理（Ｓ６０～Ｓ６３、Ｓ６５、Ｓ６７およびＳ６９）は、図１９を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０１７１】

図４６は、配置された命令の一例を示す図である。Ｎサイクル目には、第１スロット、第２スロットおよび第３スロットで実行される命令として、“ｌｄ　Ｒ０，Ｒ１”、“ｓｕｂ１　Ｒ２，Ｒ３”および“ａｄｄ１　Ｒ４，Ｒ５”がそれぞれ配置されているものとする。Ｎ＋１サイクル目には、第１スロット、第２スロットおよび第３スロットで実行される命令として、“ｓｔ　Ｒ５，Ｒ８”、“ｍｕｌ　Ｒ２，Ｒ３”および“ｍｏｄ　Ｒ０，Ｒ１０”がそれぞれ配置されているものとする。

【０１７２】

図４７は、命令並び作成処理（Ｓ６１）を説明するための図である。たとえば、図４６に示すＮ＋１サイクル目に配置された３つの命令より、図４７（ａ）～（ｆ）に示す６つの命令並びが作成される。

【０１７３】

図４８は、レジスタフィールド個数算出処理（Ｓ２４２）を説明するための図である。たとえば、図４６に示すＮサイクル目の命令並びと、図４７（ｆ）に示すＮ＋１サイクル目の命令並びとの間で、スロットごとに同一のレジスタ番号を有するレジスタフィールドの個数を求める。第１スロットについては、レジスタＲ０が両サイクルの同一レジスタフィールドで共通し、その他レジスタフィールドのレジスタは異なるため、当該個数は１である。第２スロットについては、レジスタＲ２およびＲ３が両サイクルの同一レジスタフィールドで共通するため、当該個数は２である。第３スロットについては、同一レジスタフィールドで共通するレジスタがないため、当該個数は０である。

【０１７４】

したがって、図４８の例における同一のレジスタ番号を有するレジスタフィールドの個数の和は３である。レジスタフィールド個数和算出処理（Ｓ２４４）では、このようにして、図４６に示すＮサイクル目の命令並びと、図４７（ａ）～（ｆ）に示す６通りの命令並びとの各々との間でレジスタフィールドの個数の和が求められ、それぞれ０，２，０，０，０，１および３となる。命令並び選択処理（Ｓ２４６）では、６通りの命令並びのうち、最大のレジスタフィールドの個数和をとる図４７（ｆ）に示す命令並びが選択される。

【０１７５】

なお、本変形例のレジスタフィールドの個数を求める処理（Ｓ２４２）は、レジスタが割付けられていることが前提となっている。このため、本変形例のサイクル内配置調整処理部２３７での処理は、レジスタが割付けられていない命令スケジューリング部２３２での処理の後に実行することはできず、レジスタ割付部２３４での処理の後または命令再スケジューリング部２３６での処理の後に実行される。

【０１７６】

[サイクル内配置調整処理部２３７の第４変形例]

サイクル内配置調整処理部２３７は、図１９を参照して説明した処理の代わりに、図４９に示す処理を実行してもよい。

【０１７７】

図４９は、サイクル内配置調整処理部２３７の動作の第４変形例を示すフローチャートである。

サイクル内配置調整処理部２３７は、図１９の命令並びごとにハミング距離の和を求める処理（Ｓ６３～Ｓ６６）の代わりに、以下の処理を実行する。すなわち、サイクル内配置調整処理部２３７は、着目している命令並びに含まれる命令のうち、スロットのデフォルト論理に合致する命令の個数を求める（Ｓ２５２）。

【０１７８】

また、サイクル内配置調整処理部２３７は、図１９の命令の並べ替え処理（Ｓ６８）の代わりに、以下の処理を実行する。すなわち、サイクル内配置調整処理

部２３７は、６通りの命令並びの各々について求められたデフォルト論理に合致する命令の個数のうち、最大個数をとる命令並びを選択し、その命令並びの並びになるように命令の並べ替えを行なう（Ｓ２５４）。その他の処理（Ｓ６０～Ｓ６２、Ｓ６７およびＳ６９）は、図１９を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０１７９】

たとえば、命令並び作成処理（Ｓ６１）で図４７（ａ）～（ｆ）に示す６つの命令並びが作成されたものとする。上述のとおり、命令並びに含まれる各命令が配置されたスロットのデフォルト論理に合致するか否かは、命令の先頭２ビットを参照することにより判別可能である。たとえば、図４７（ｂ）に示す命令並びでは、第１スロットに配置された命令の先頭２ビットが“０１”であるため、当該スロットのデフォルト論理に合致するが、第２スロットおよび第３スロットに配置された命令の先頭２ビットはそれぞれ“１１”および“１０”であるため、当該スロットのデフォルト論理には合致しない。このため、当該スロットでデフォルト論理に合致する命令は１つである。このようにして、個数算出処理（Ｓ２５２）では６通りの命令並びの各々についてデフォルト論理に合致する命令の個数が求められ、それぞれ３、１、１、０、０および１となる。命令並び選択処理（Ｓ２５４）では、６通りの命令並びのうち、デフォルト論理に合致する命令数の最大値をとる図４７（ａ）に示す命令並びが選択される。

【０１８０】

以上のように、本実施の形態におけるコンパイラ２００によれば、同一スロットのサイクル間で命令、オペコードおよびレジスタフィールドのハミング距離が小さくなるように命令の配置の最適化が行なわれる。このため、プロセッサの命令レジスタに保持される値のビット変化が小さく、プロセッサを低消費電力で動作させることができる機械語プログラムが生成される。

【０１８１】

また、同一スロットの同一レジスタフィールドにおいて、同一のレジスタのアクセスが連続するような命令の配置の最適化が行なわれる。このため、同一のレジスタへのアクセスが連続し、レジスタを選択するための制御信号の変化が少な

くなり、プロセッサを低消費電力で動作させることができる機械語プログラムが生成される。

【０１８２】

さらに、各スロットにはデフォルト論理に合致するように命令が割り当てられる。このため、同一のスロットでは、プロセッサの同じ構成要素を利用する命令が連続して実行されることになる。このため、プロセッサを低消費電力で動作させることができる機械語プログラムが生成される。

【０１８３】

さらにまた、１スロットまたは２スロットしか使用しない命令の命令サイクルが連続した場合には、その間、空きスロットへの電力の供給を停止させることができる。このため、プロセッサを低消費電力で動作させることができる機械語プログラムが生成される。

【０１８４】

さらにまた、プラグマまたはコンパイル時のオプションでプログラム実行時に使用するスロット数を指定することができる。このため、空きスロットを発生させることができ、空きスロットへの電力の供給を停止させることができる。このため、プロセッサを低消費電力で動作させることができる機械語プログラムが生成される。

【０１８５】

以上、本発明に係るコンパイラについて、実施の形態に基づいて説明したが、本発明は、この実施の形態に限定されるものではない。

例えば、図２８および図２９を参照して説明した命令再スケジューリング部２３２の最適命令取出し処理（Ｓ１２２）では、同一のレジスタ番号を有するフィールドの個数（Ｓ１５２）、直前に実行される命令とのハミング距離（Ｓ１５６）、スロットのデフォルト論理（Ｓ１６０）の順に優先して最適命令を特定したが、この優先順位はこれに限られるものではなく、その他の優先順位で最適命令を特定してもよい。

【０１８６】

また、最適命令を特定する際に考慮するハミング距離、スロットのデフォルト

論理等の各種条件も、この実施の形態に限定されるものではない。要するに、本発明に係るコンパイラによりプロセッサが動作した際に、総消費電力量が小さくなるような、条件の組み合わせや、優先順位であればよいのである。なお、命令スケジューリング部２３２以外の、命令スケジューリング部２３２、レジスタ割付部２３４およびサイクル内配置調整処理部２３７などの処理においても同様であるのは言うまでもない。

【０１８７】

さらに、これら条件の組み合わせや、優先順位をパラメータ化し、ソースプログラム２０２のヘッダファイルとしてコンパイル時に組み込む構成としてもよいし、これらのパラメータをコンパイラのオプションとして指定可能としてもよい。

【０１８８】

さらにまた、本実施の形態の最適化部２３０での処理は、基本ブロックごとにいくつかのスケジューリング方法の中から最適なものを選択するようにしてもよい。例えば、基本ブロックごとに、あらかじめ用意された複数のスケジューリング方法のすべてについてスケジュール結果を求め、最も消費電力が小さくなると予測されるスケジューリング方法を選択するようにしてもよい。

【０１８９】

また、バックトラックなどの手法を用いて最適なスケジューリング方法を選択するようにしてもよい。例えば、命令スケジューリング部２３２において最も消費電力が小さくなると予測されるスケジューリング方法を選択した後であっても、レジスタ割付部２３４でレジスタ割付を行なったところ、予測消費電力が予定していた値よりも大きくなった場合には、命令スケジューリング部２３２において２番目に消費電力が小さくなると予測されるスケジューリング方法を選択して、レジスタ割付を行なってみる。その結果、予測消費電力が予定していた値よりも小さくなれば、命令再スケジューリング部２３６による命令再スケジュール処理を実行するようにしてもよい。　さらにまた、本実施の形態では、Ｃ言語で記述されたソースプログラムを機械語プログラムに変換する例について説明したが、ソースプログラムはＣ言語以外の高級言語であってもよいし、他のコンパイラ

ですでにコンパイルされた機械語プログラムであってもよい。ソースプログラムが機械語プログラムの場合には、その機械語プログラムを最適化した機械語プログラムが出力される構成となる。

【０１９０】

【発明の効果】

以上の説明から明らかように、本発明に係るコンパイラ装置によると、プロセッサの命令レジスタに保持される値のビット変化が小さく、プロセッサを低消費電力で動作させることができる命令列が生成される。

【０１９１】

また、同一のレジスタへのアクセスが連続し、レジスタを選択するための制御信号の変化が少なくなり、プロセッサを低消費電力で動作させることができる命令列が生成される。

【０１９２】

さらに、同一スロットで同じ構成要素を利用する命を連続して実行させることができるため、プロセッサを低消費電力で動作させることができる命令列が生成される。

【０１９３】

さらにまた、空きスロットへの電力の供給を停止させることができるため、プロセッサを低消費電力で動作させることができる命令列が生成される。

以上のように、本発明に係るコンパイラにより、並列処理可能なプロセッサを低消費電力で動作させることが可能となる。特に、携帯電話、情報携帯端末等の携帯型の情報処理装置のように、低消費電力での動作が求められる装置で用いられるプロセッサ向きの命令列（機械語プログラム）を生成することができ、その実用的価値は極めて高い。

【図面の簡単な説明】

【図１】　本実施の形態に係るプロセッサが解読実行する命令の構造を示す図である。

【図２】　本実施の形態に係るプロセッサの概略構成を示すブロック図である。

【図３】　パケットの一例を示す図である。

【図４】　パケットに含まれる並列実行境界情報について説明するための図である。

【図５】　パケットの並列実行境界情報に基づいて作成された並列実行される命令の実行単位の一例を示す図である。

【図６】　算術論理・比較演算部の概略構成を示すブロック図である。

【図７】　バレルシフタの概略構成を示すブロック図である。

【図８】　除算器の概略構成を示すブロック図である。

【図９】　乗算・積和演算部の概略構成を示すブロック図である。

【図１０】　プロセッサによる命令実行時の各パイプライン動作を示すタイミング図である。

【図１１】　プロセッサで実行される命令、処理の内容およびそのビットパターンを示す図である。

【図１２】　本実施の形態に係るコンパイラの構成を示す機能ブロック図である。

【図１３】　命令スケジューリング部の動作を示すフローチャートである。

【図１４】　依存グラフの一例を示す図である。

【図１５】　命令のスケジュール結果の一例を示す図である。

【図１６】　図１３の最適命令取出し処理の動作を示すフローチャートである。

【図１７】　オペコードのビットパターン間のハミング距離の算出方法を説明するための図である。

【図１８】　ビット長が異なるオペコード間でのハミング距離の算出方法を説明するための図である。

【図１９】　サイクル内配置調整処理部の動作を示すフローチャートである。

【図２０】　６通りの命令並びの一例を示す図である。

【図２１】　配置された命令の一例を示す図である。

【図２２】　命令並び作成処理（図１９のＳ６１）を説明するための図であ

る。

【図２３】　オペコードのハミング距離算出処理（図１９のＳ６４）を説明するための図である。

【図２４】　レジスタ割付部の動作を示すフローチャートである。

【図２５】　割付対象となる変数の生存区間を示す図である。

【図２６】　図２５の例に基づいて作成された変数の干渉グラフを示す図である。

【図２７】　命令スケジューリングの途中結果を示す図である。

【図２８】　命令再スケジューリング部の動作を示すフローチャートである。

【図２９】　図２８の最適命令取出し処理の動作を示すフローチャートである。

【図３０】　配置候補命令特定処理（図２９のＳ１５２）を説明するための図である。

【図３１】　配置候補命令特定処理（図２９のＳ１５６）を説明するための図である。

【図３２】　配置候補命令特定処理（図２９のＳ１６０）を説明するための図である。

【図３３】　スロット停止・復帰命令生成部の動作を示すフローチャートである。

【図３４】　命令が配置されたスケジュール結果の一例を示す図である。

【図３５】　図３３の特定の１スロットのみが連続使用されている場合の処理で命令が書き込まれたスケジュール結果の一例を示す図である。

【図３６】　図３３の特定の２スロットのみが連続使用されている場合の処理で命令が書き込まれたスケジュール結果の一例を示す図である。

【図３７】　プログラム状態レジスタの一例を示す図である。

【図３８】　プログラム状態レジスタの他の一例を示す図である。

【図３９】　図２８の最適命令取出し処理の他の動作を示すフローチャートである。

【図４０】　配置候補命令特定処理（図３９のＳ２１２）を説明するための図である。

【図４１】　サイクル内配置調整処理部２３７の動作の第１変形例を示すフローチャートである。

【図４２】　命令のハミング距離算出処理（図４１のＳ２２２）を説明するための図である。

【図４３】　サイクル内配置調整処理部２３７の動作の第２変形例を示すフローチャートである。

【図４４】　レジスタフィールドのハミング距離算出処理（図４３のＳ２３２）を説明するための図である。

【図４５】　サイクル内配置調整処理部２３７の動作の第３変形例を示すフローチャートである。

【図４６】　配置された命令の一例を示す図である。

【図４７】　命令並び作成処理（図４５のＳ６１）を説明するための図である。

【図４８】　レジスタフィールド個数算出処理（図４５のＳ２４２）を説明するための図である。

【図４９】　サイクル内配置調整処理部２３７の動作の第４変形例を示すフローチャートである。

【符号の説明】

３０　　プロセッサ

４０　　命令メモリ

５０　　命令供給発行部

５２　　命令フェッチ部

５４　　命令バッファ

５６　　命令レジスタ部

５６ａ～５６ｃ　　命令レジスタ

６０　　解読部

６２　　命令発行制御部

６４　　デコード部

６４ａ～６４ｃ　　命令デコーダ

７０　　実行部

７２　　実行制御部

７４　　ＰＣ部

７６　　レジスタファイル

７８ａ～７８ｃ　　算術論理・比較演算部

８０ａ，８０ｂ　　乗算・積和演算部

８２ａ～８２ｃ　　バレルシフタ

８４　　除算器

８８　　オペランドアクセス部

９０，９２　　データバス

９４　　ＯＤバス

９６　　ＯＡバス

１００　　データメモリ

１１２，１１４　　パケット

１２２，１２４，１２６　　実行単位

１３２　　ＡＬＵ部

１３４，１５４，１８０　　飽和処理部

１３６　　フラグ部

１４２，１６２，１７２　　アキュムレータ部

１４６，１４８，１７８　　セレクタ

１５０　　上位バレルシフタ

１５２　　下位バレルシフタ

１６４　　除算部

１７４ａ，１７４ｂ　　３２ビット乗算器（MUL）

１７６ａ～１７６ｃ　　６４ビット加算器（Adder）

２００　　コンパイラ

２０２　　ソースプログラム

２０４　機械語プログラム

２１０　パーサー部

２２０　中間コード変換部

２３０　最適化部

２３２　命令スケジューリング部

２３４　レジスタ割付部

２３６　命令再スケジューリング部

２３７　サイクル内配置調整処理部

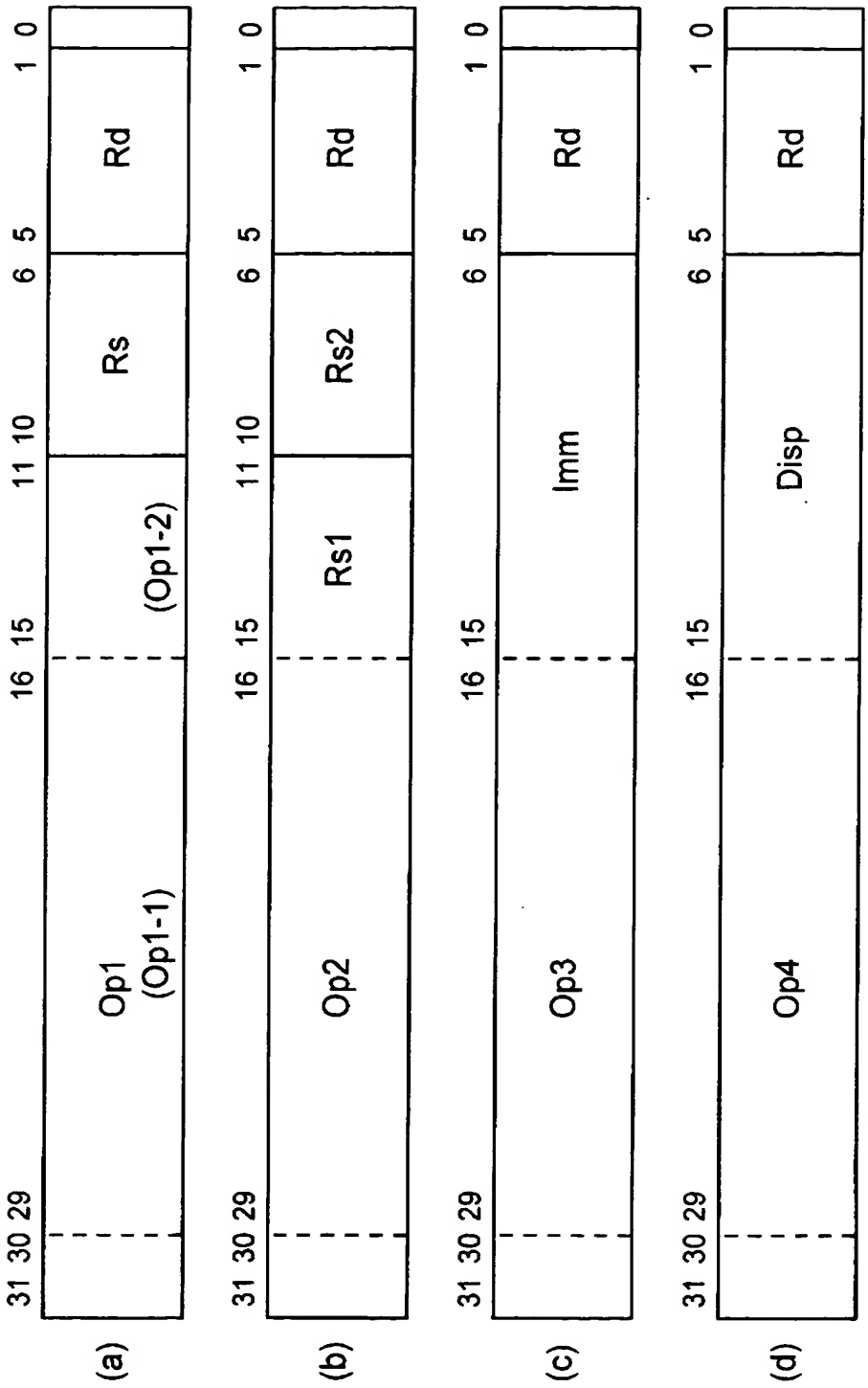
２３８　スロット停止・復帰命令生成部

２３９　並列実行境界情報設定部

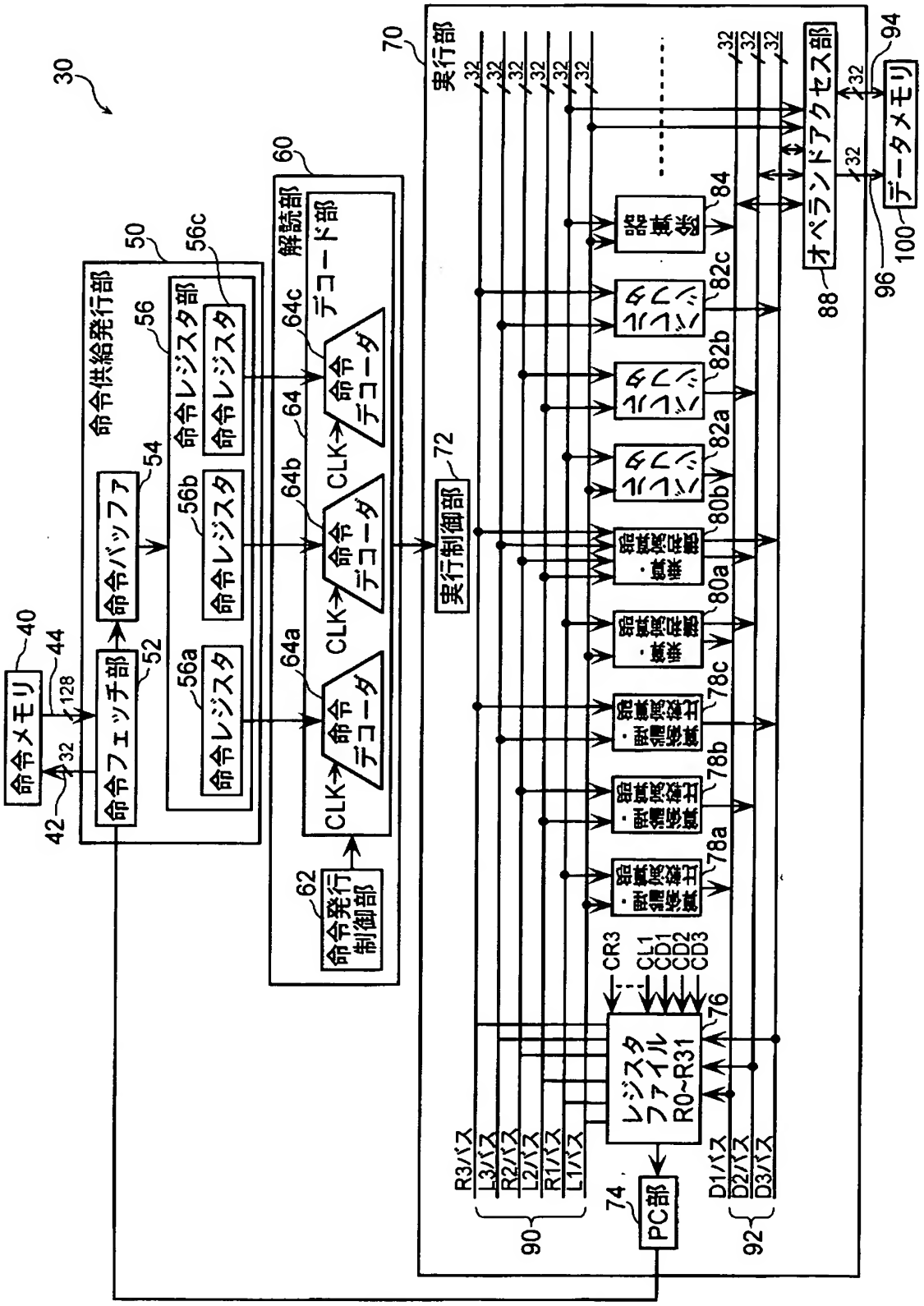
２４０　コード生成部

【書類名】　図面

【図１】


31 30 29
16 15
11 10
6 5
1 0
Op1
(Op1-1)
(Op1-2)
Rs
Rd
(a)
Op2
Rs1
Rs2
Rd
(b)
Op3
Imm
Rd
(c)
Op4
Disp
Rd
(d)

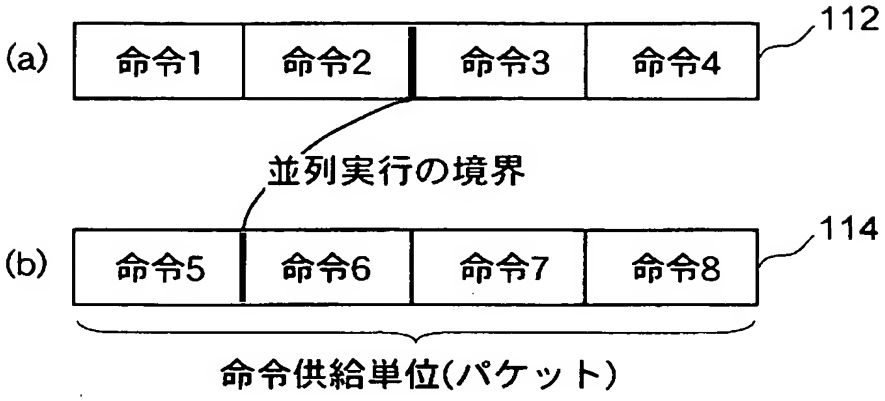
【図２】


30
命令メモリ
40
42
44
128
32
命令フェッチ部
52
命令供給発行部
50
命令バッファ
54
命令レジスタ部
56
56a
命令レジスタ
56b
命令レジスタ
56c
命令レジスタ
解読部
60
デコード部
64
64a
64b
64c
CLK
命令デコーダ
命令発行制御部
62
実行制御部
72
実行部
70
32
CR3
CL1
CD1
CD2
CD3
レジスタファイル R0~R31
76
PC部
74
R3バス
L3バス
R2バス
L2バス
R1バス
L1バス
90
D1バス
D2バス
D3バス
92
算術・論理演算器
84
82a
82b
82c
80a
80b
78a
78b
78c
オペランドアクセス部
88
データメモリ
94
96
100

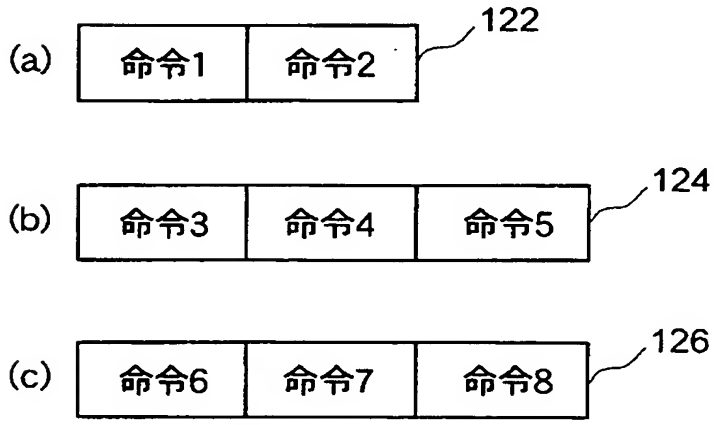
【図３】


命令1
命令2
命令3
命令4

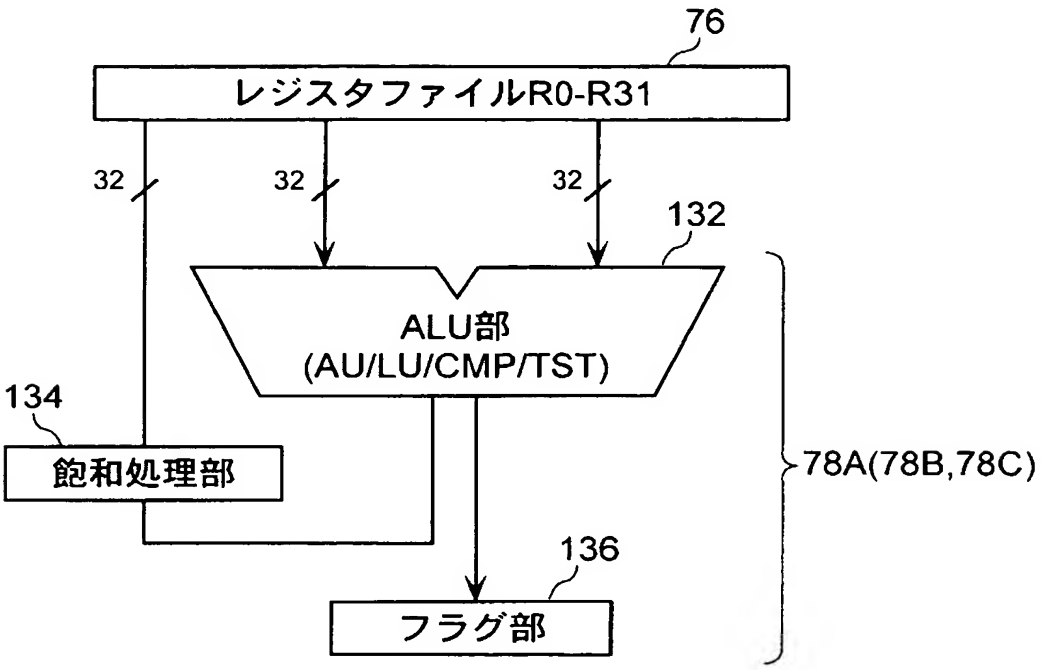
【図４】


(a)
命令1
命令2
命令3
命令4
112
並列実行の境界
(b)
命令5
命令6
命令7
命令8
114
命令供給単位(パケット)

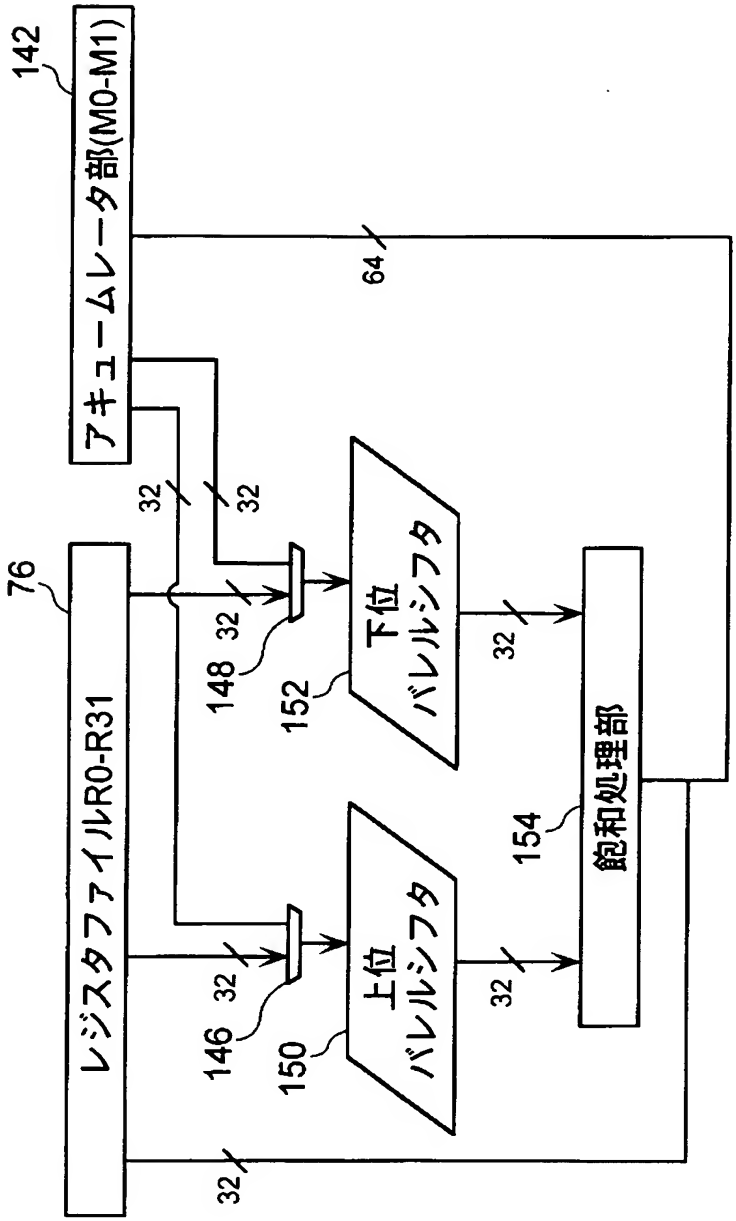
【図５】


(a)
命令1
命令2
122
(b)
命令3
命令4
命令5
124
(c)
命令6
命令7
命令8
126

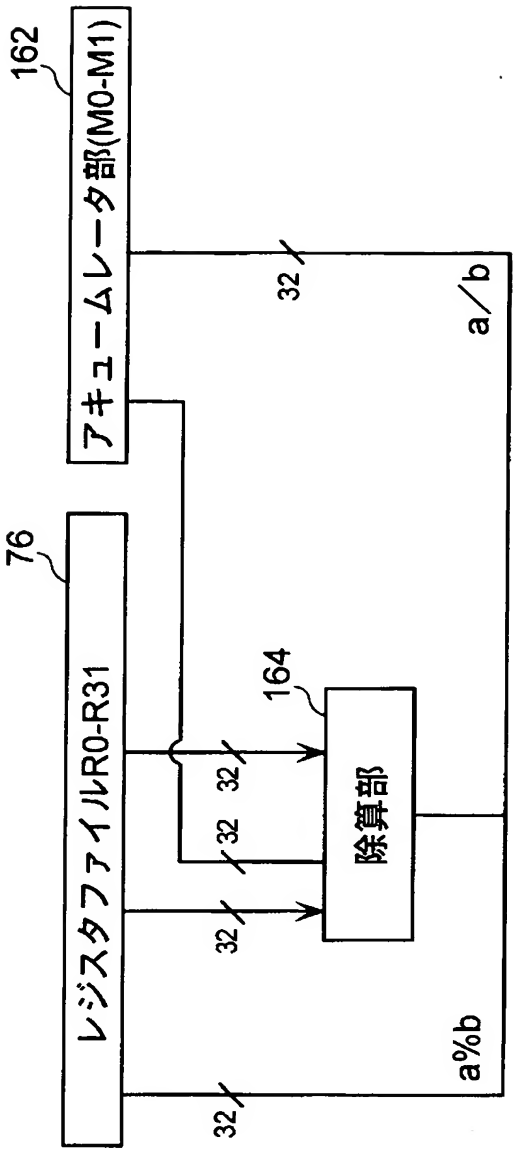
【図６】


76
レジスタファイルR0-R31
32
32
32
132
ALU部
(AU/LU/CMP/TST)
134
飽和処理部
78A(78B,78C)
136
フラグ部

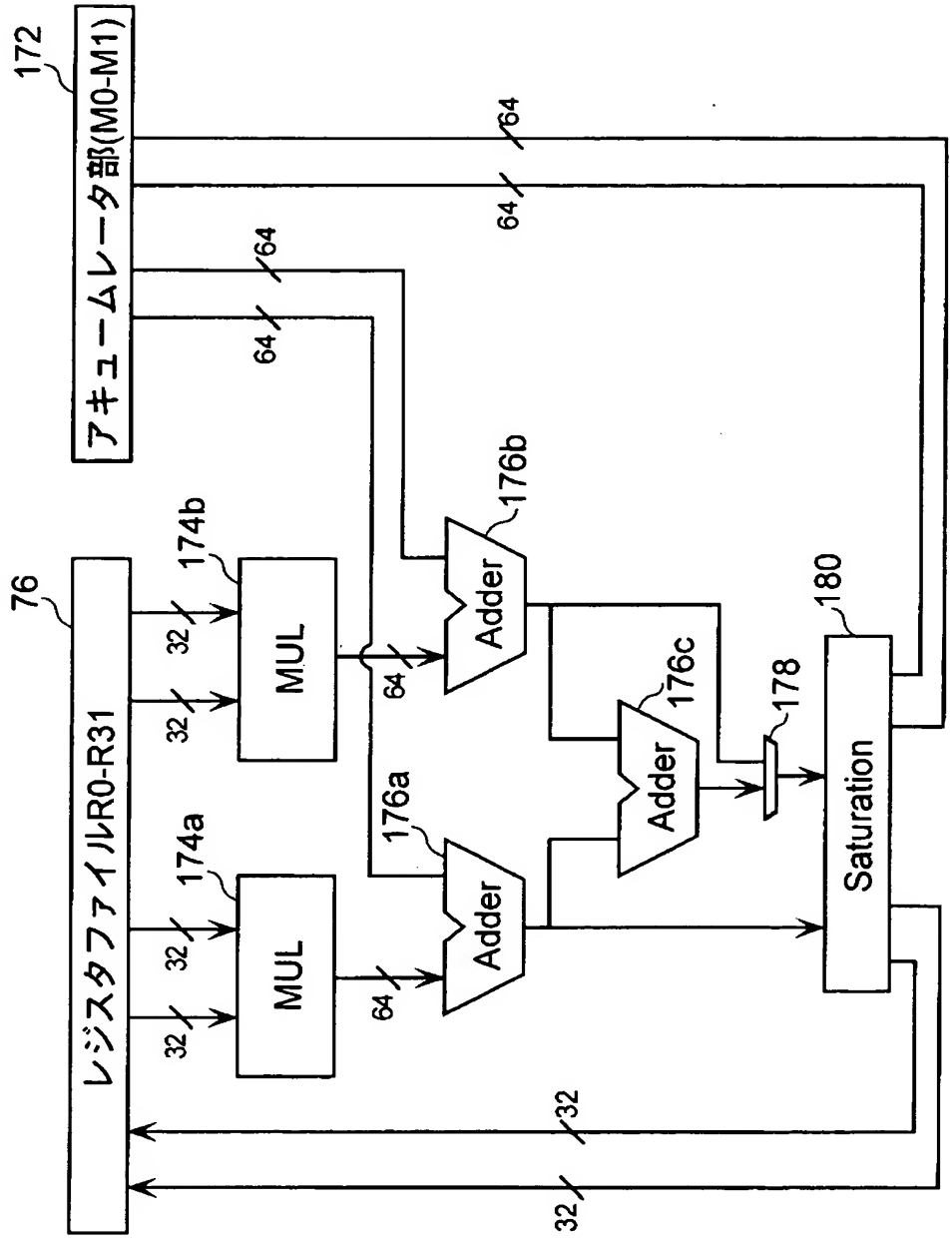
【図７】


142
アキュームレータ部(M0-M1)
64
32
32
76
レジスタファイルR0-R31
32
148
152
下位
バレルシフタ
32
154
飽和処理部
32
146
150
上位
バレルシフタ
32
32

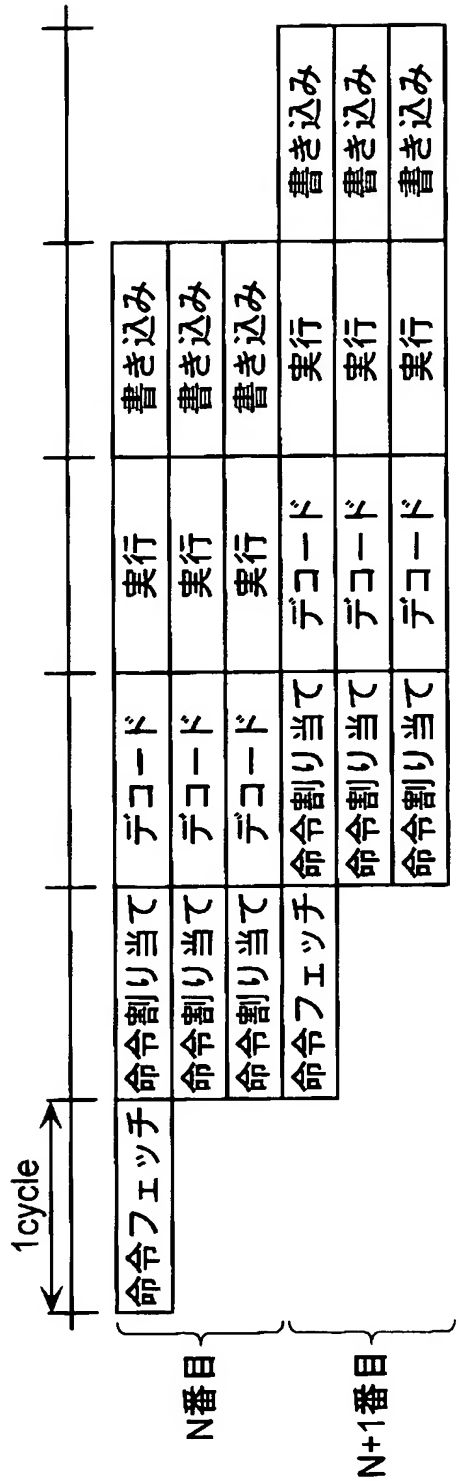
【図８】


162
アキュームレータ部(M0-M1)
32
a／b
76
レジスタファイルR0-R31
164
32
32
除算部
32
a%b
32

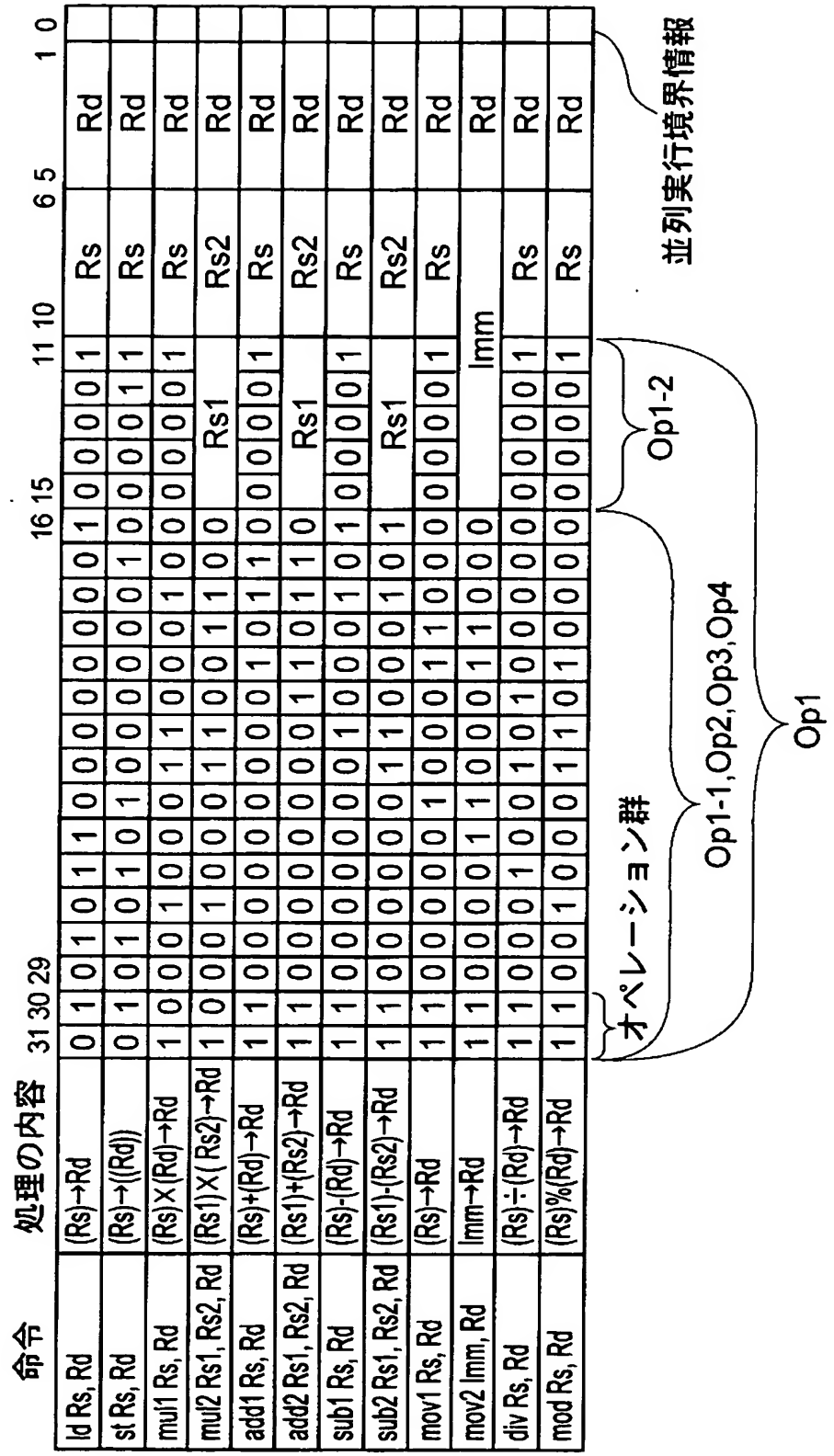
【図９】


172
アキュームレータ部(M0-M1)
64
64
64
64
76
レジスタファイルR0-R31
174b
174a
MUL
MUL
32
32
32
32
176b
176a
176c
Adder
Adder
Adder
64
64
178
180
Saturation
32
32

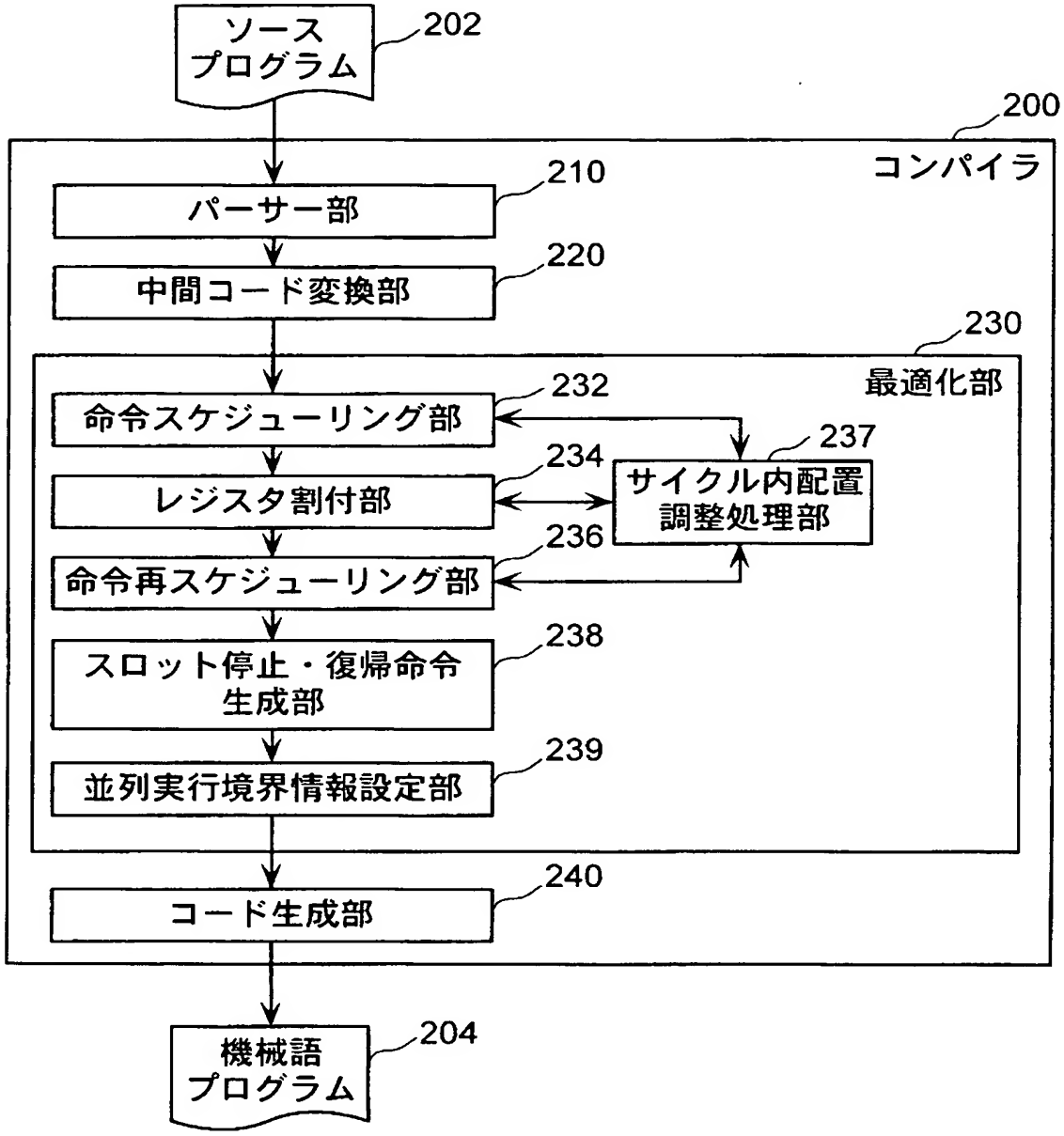
【図１０】


1cycle
命令フェッチ
命令割り当て
デコード
実行
書き込み
命令フェッチ
命令割り当て
デコード
実行
書き込み
命令フェッチ
命令割り当て
デコード
実行
書き込み
命令フェッチ
命令割り当て
デコード
実行
書き込み
命令フェッチ
命令割り当て
デコード
実行
書き込み
N番目
N+1番目

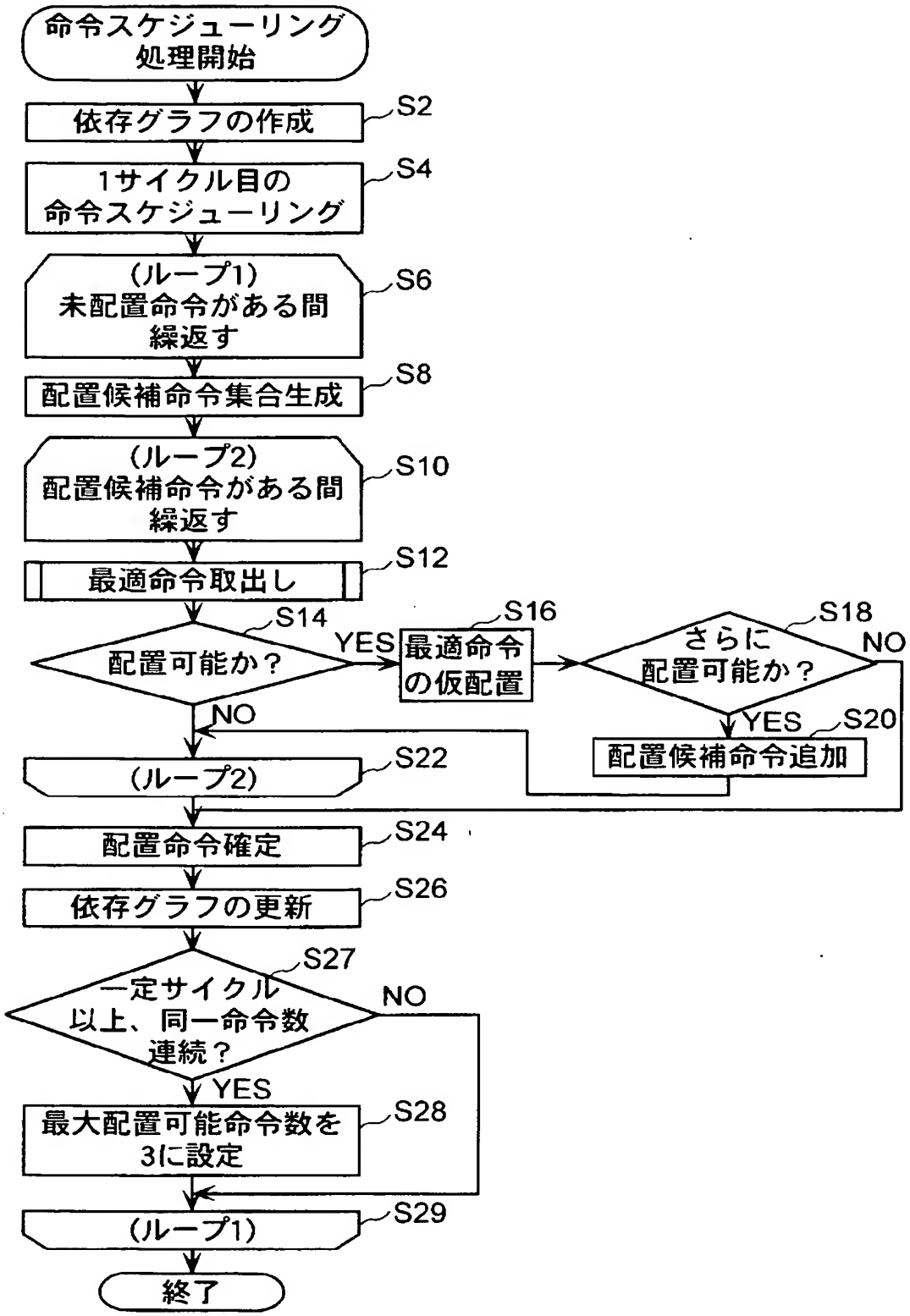
【図１１】


命令
処理の内容
31 30 29
16 15
11 10
6 5
1 0
ld Rs, Rd
(Rs)→Rd
st Rs, Rd
(Rs)→((Rd))
mul1 Rs, Rd
(Rs)X(Rd)→Rd
mul2 Rs1, Rs2, Rd
(Rs1)X(Rs2)→Rd
add1 Rs, Rd
(Rs)+(Rd)→Rd
add2 Rs1, Rs2, Rd
(Rs1)+(Rs2)→Rd
sub1 Rs, Rd
(Rs)-(Rd)→Rd
sub2 Rs1, Rs2, Rd
(Rs1)-(Rs2)→Rd
mov1 Rs, Rd
(Rs)→Rd
mov2 Imm, Rd
Imm→Rd
div Rs, Rd
(Rs)÷(Rd)→Rd
mod Rs, Rd
(Rs)%(Rd)→Rd
Rs
Rs1
Rs2
Imm
Rd
オペレーション群
Op1-1,Op2,Op3,Op4
Op1-2
Op1
並列実行境界情報

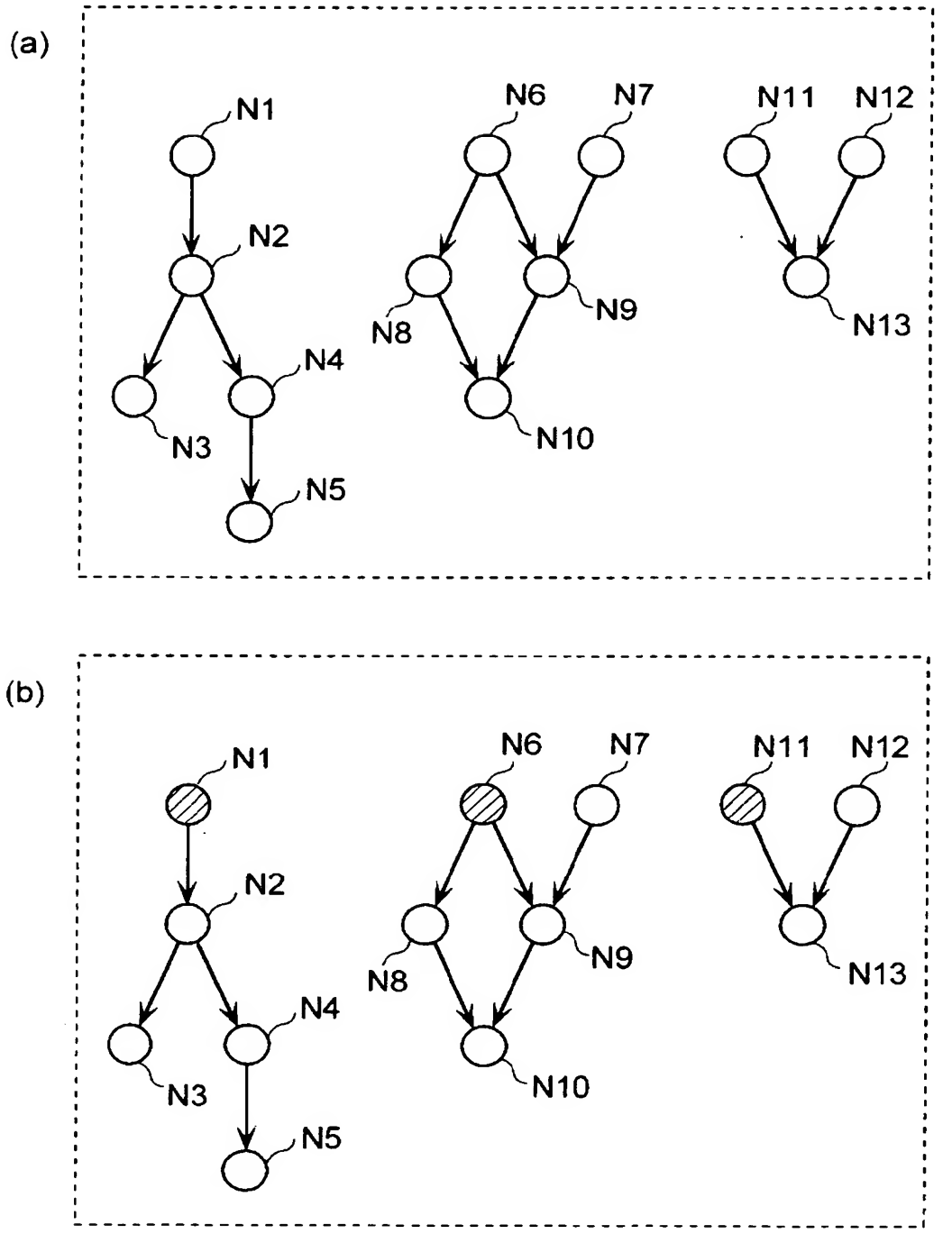
【図１２】


ソース
プログラム
202
200
コンパイラ
210
パーサー部
220
中間コード変換部
230
最適化部
232
命令スケジューリング部
237
234
レジスタ割付部
サイクル内配置
調整処理部
236
命令再スケジューリング部
238
スロット停止・復帰命令
生成部
239
並列実行境界情報設定部
240
コード生成部
機械語
プログラム
204

【図１３】


命令スケジューリング処理開始
依存グラフの作成
S2
1サイクル目の命令スケジューリング
S4
(ループ1) 未配置命令がある間繰返す
S6
配置候補命令集合生成
S8
(ループ2) 配置候補命令がある間繰返す
S10
最適命令取出し
S12
配置可能か？
S14
YES
最適命令の仮配置
S16
さらに配置可能か？
S18
NO
YES
配置候補命令追加
S20
NO
(ループ2)
S22
配置命令確定
S24
依存グラフの更新
S26
一定サイクル以上、同一命令数連続？
S27
NO
YES
最大配置可能命令数を3に設定
S28
(ループ1)
S29
終了

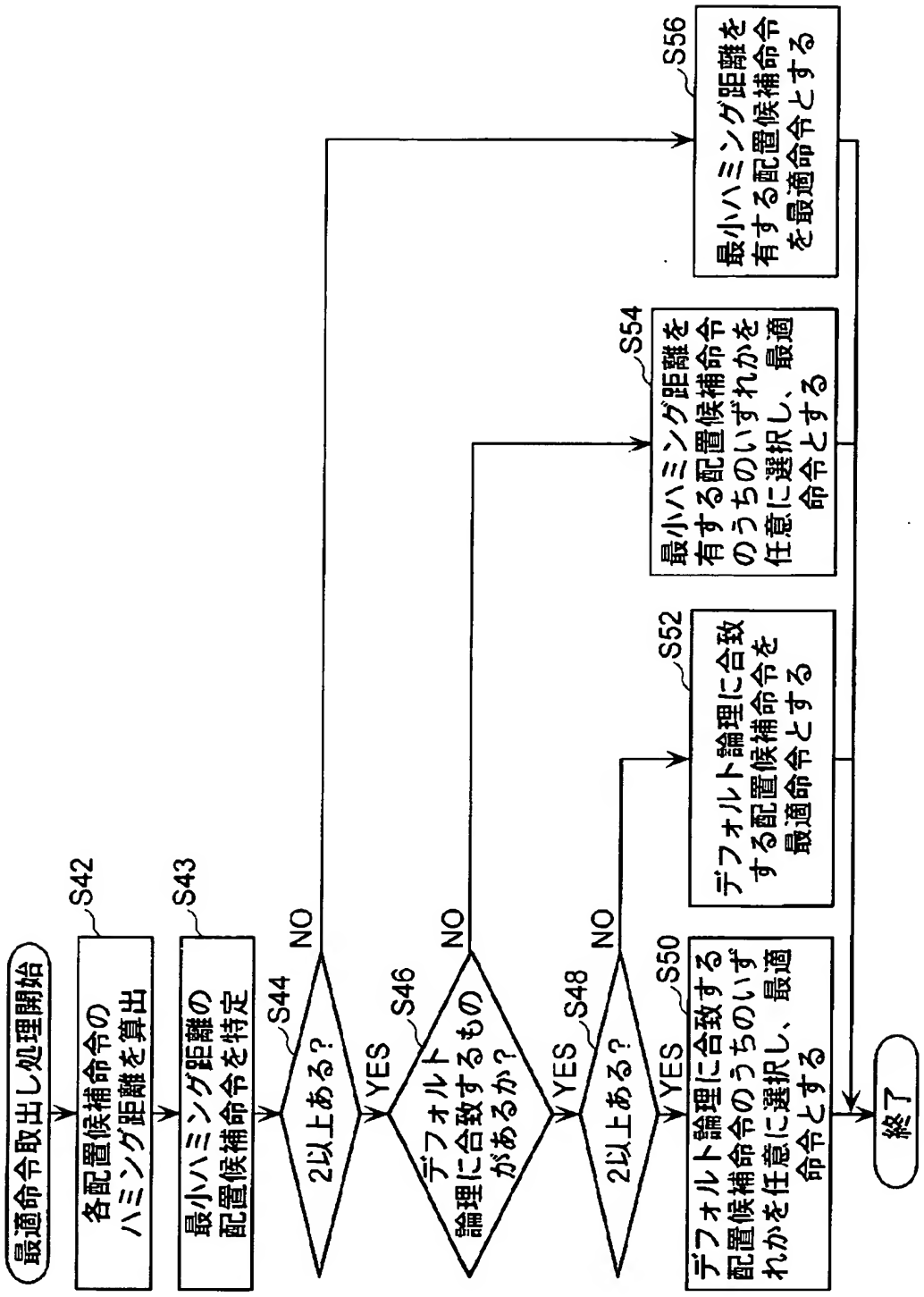
【図１４】


(a)
N1
N2
N3
N4
N5
N6
N7
N8
N9
N10
N11
N12
N13
(b)
N1
N2
N3
N4
N5
N6
N7
N8
N9
N10
N11
N12
N13
：配置済ノード
：未配置ノード

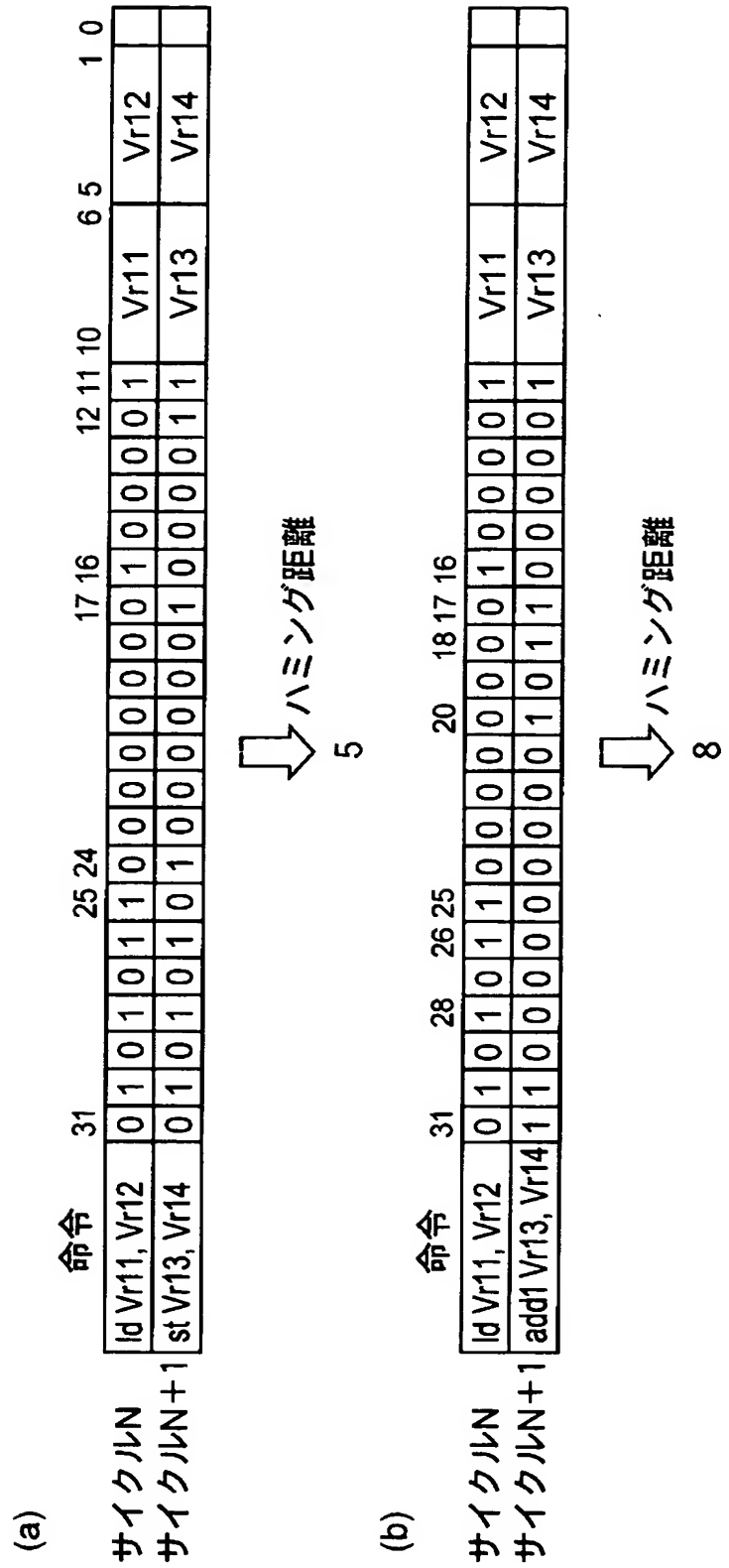
【図１５】

サイクル	第1スロット	第2スロット	第3スロット
1	ld Vr0, Vr1 (ノードN1)	mul2 Vr2, Vr3, Vr4 (ノードN11)	add1 Vr5, Vr6 (ノードN6)

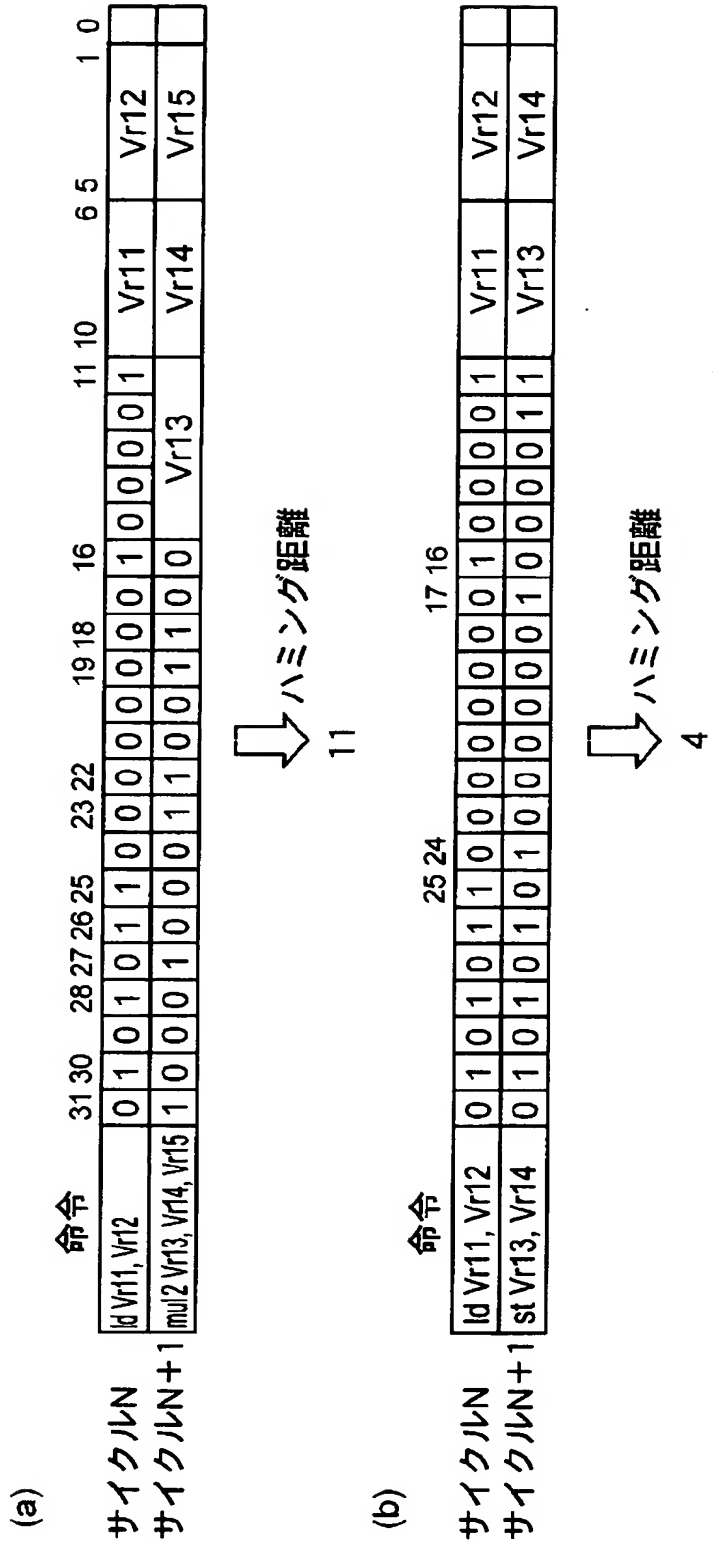
【図１６】


最適命令取出し処理開始
S42
各配置候補命令の
ハミング距離を算出
S43
最小ハミング距離の
配置候補命令を特定
S44
2以上ある？
NO
YES
S46
デフォルト
論理に合致するもの
があるか？
NO
YES
S48
2以上ある？
NO
YES
S50
デフォルト論理に合致する
配置候補命令のうちのいず
れかを任意に選択し、最適
命令とする
S52
デフォルト論理に合致
する配置候補命令を
最適命令とする
S54
最小ハミング距離を
有する配置候補命令
のうちのいずれかを
任意に選択し、最適
命令とする
S56
最小ハミング距離を
有する配置候補命令
を最適命令とする
終了

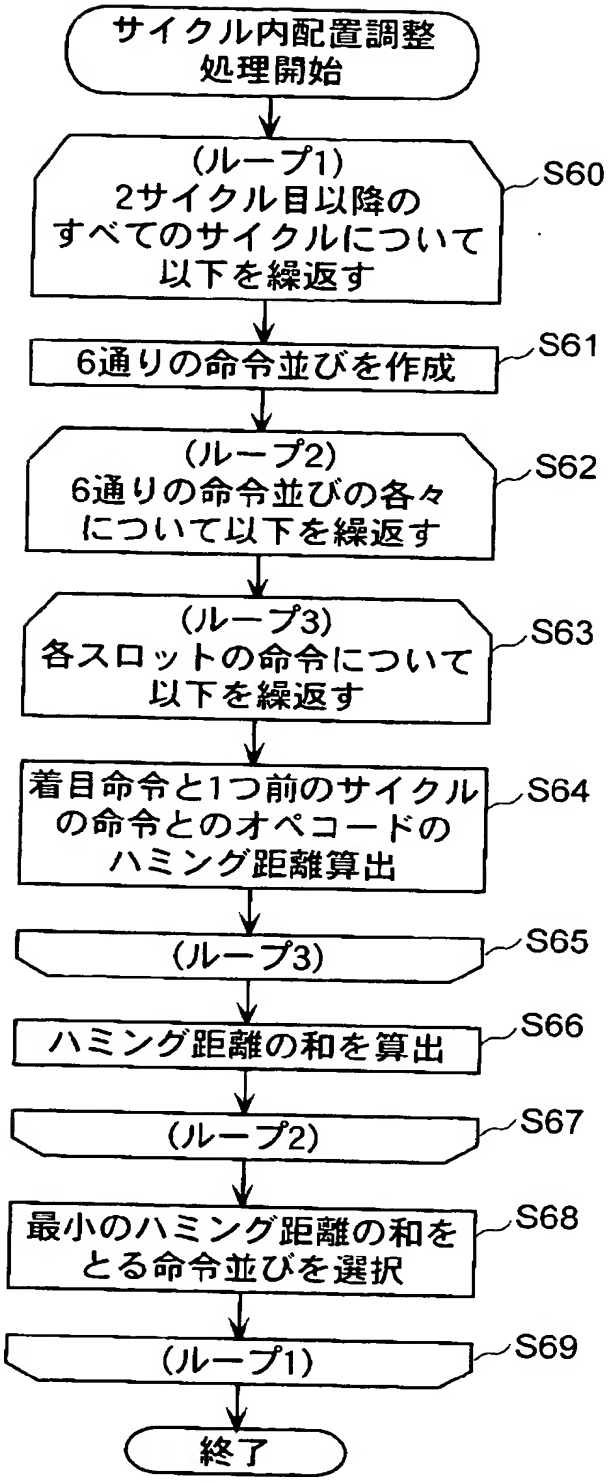
【図１７】


(a)
命令
31
25 24
17 16
12 11 10
6 5
1 0
サイクルN
ld Vr11, Vr12
Vr11
Vr12
サイクルN+1
st Vr13, Vr14
Vr13
Vr14
ハミング距離
5
(b)
命令
31
28
26 25
20
18 17 16
サイクルN
ld Vr11, Vr12
Vr11
Vr12
サイクルN+1
add1 Vr13, Vr14
Vr13
Vr14
ハミング距離
8

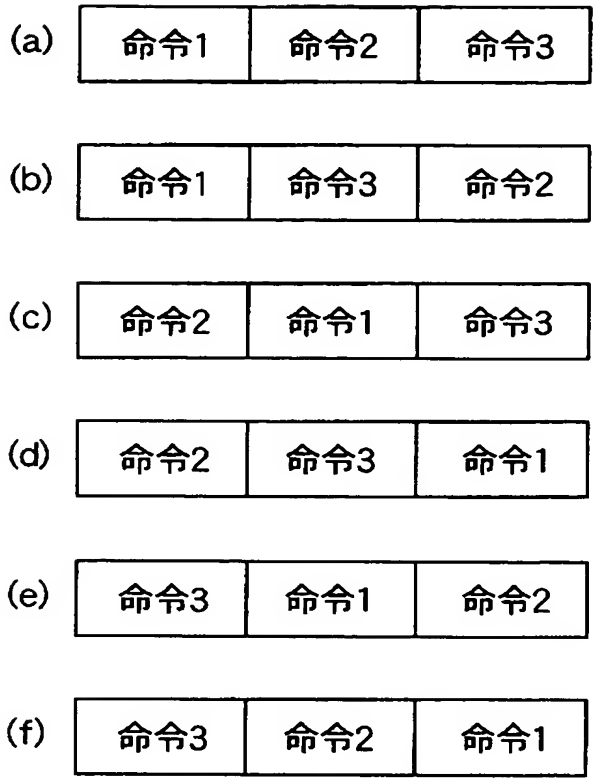
【図１８】


(a)
命令
31 30 28 27 26 25 23 22 19 18 16 11 10 6 5 1 0
サイクルN
ld Vr11, Vr12
0 1 0 1 0 1 1 0 0 0 0 0 0 1 0 0 0 0 1 Vr11 Vr12
サイクルN＋1
mul2 Vr13, Vr14, Vr15
1 0 0 0 1 0 0 0 1 1 0 0 1 1 0 0 Vr13 Vr14 Vr15
ハミング距離
11
(b)
命令
25 24 17 16
サイクルN
ld Vr11, Vr12
0 1 0 1 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 1 Vr11 Vr12
サイクルN＋1
st Vr13, Vr14
0 1 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 1 1 Vr13 Vr14
ハミング距離
4

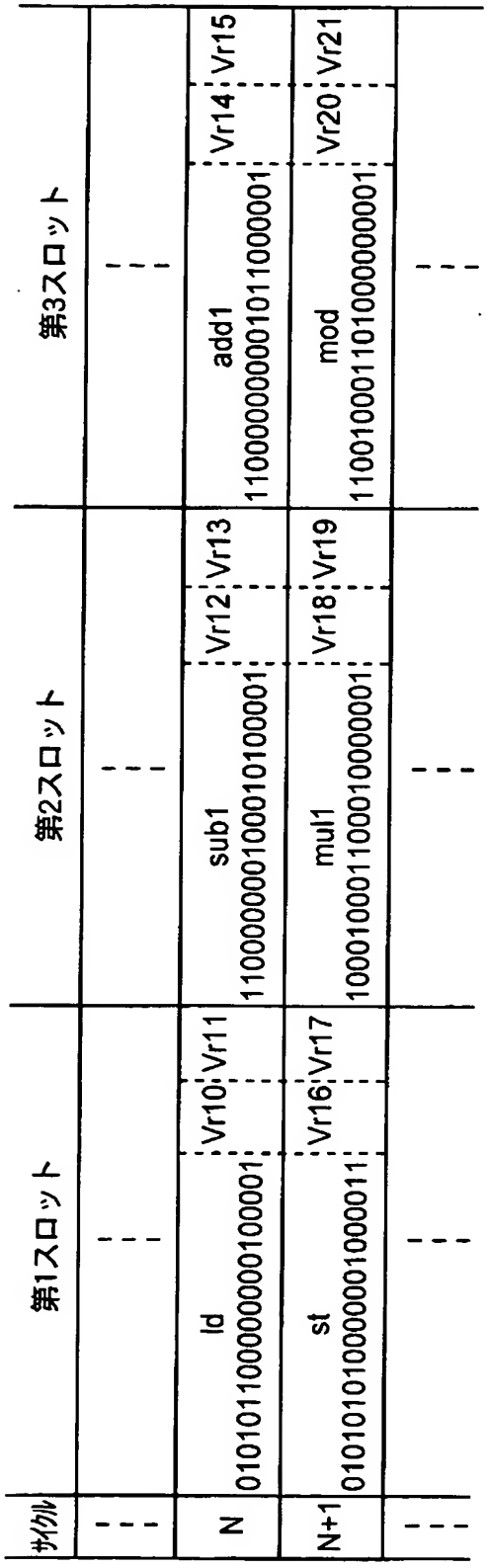
【図１９】


サイクル内配置調整
処理開始
(ループ1)
2サイクル目以降の
すべてのサイクルについて
以下を繰返す
S60
6通りの命令並びを作成
S61
(ループ2)
6通りの命令並びの各々
について以下を繰返す
S62
(ループ3)
各スロットの命令について
以下を繰返す
S63
着目命令と1つ前のサイクル
の命令とのオペコードの
ハミング距離算出
S64
(ループ3)
S65
ハミング距離の和を算出
S66
(ループ2)
S67
最小のハミング距離の和を
とる命令並びを選択
S68
(ループ1)
S69
終了

【図２０】

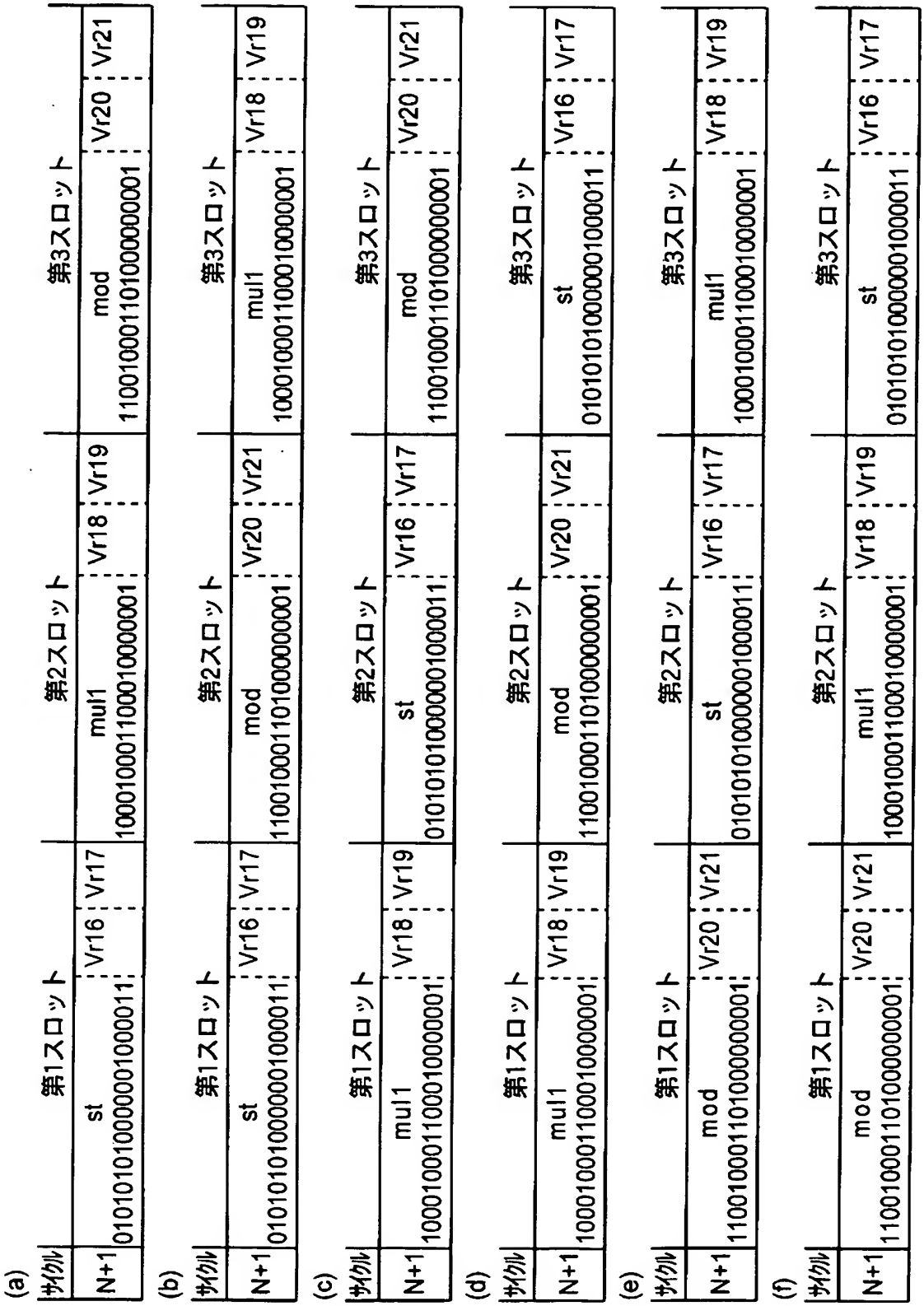

(a)
命令1
命令2
命令3
(b)
命令1
命令3
命令2
(c)
命令2
命令1
命令3
(d)
命令2
命令3
命令1
(e)
命令3
命令1
命令2
(f)
命令3
命令2
命令1

【図２１】

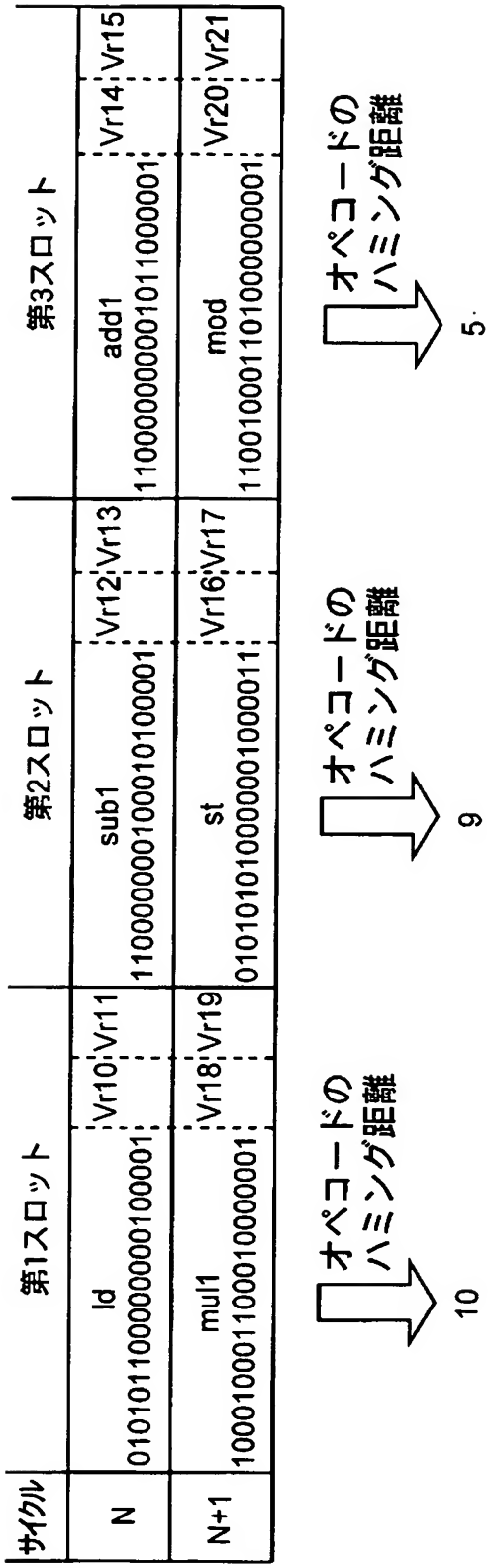

サイクル
第1スロット
第2スロット
第3スロット

N
ld
0101011000000100001
Vr10
Vr11
sub1
11000000010001010000 1
Vr12
Vr13
add1
11000000000101100000 1
Vr14
Vr15
N+1
st
0101010100000010000 11
Vr16
Vr17
mul1
10001000110001000000 1
Vr18
Vr19
mod
11001000110100000000 1
Vr20
Vr21

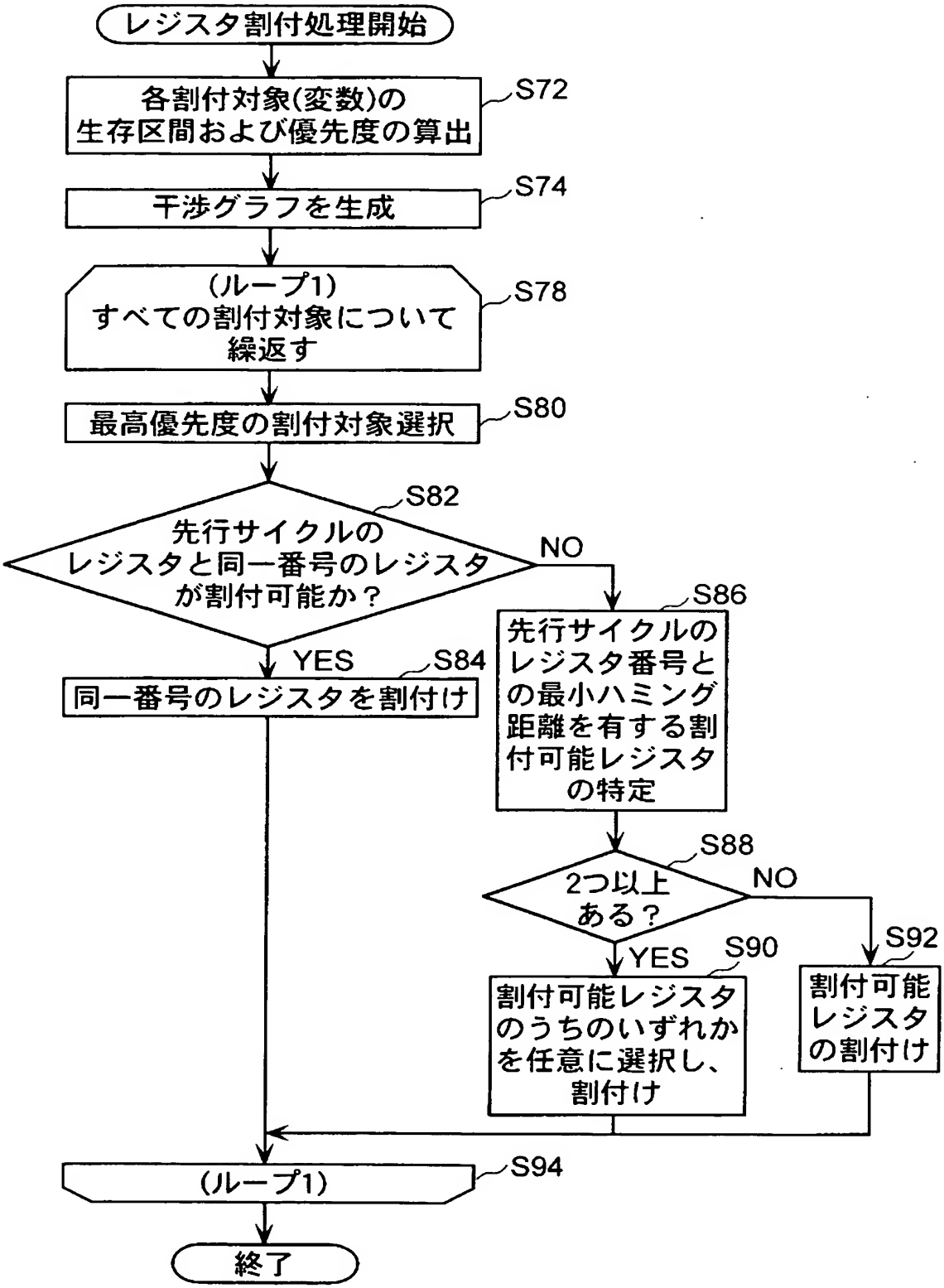
【図２２】


(a)
サイクル
第1スロット
第2スロット
第3スロット
N+1
st
0101010100000010000011
Vr16
Vr17
mul1
10001000110001000010000001
Vr18
Vr19
mod
11001000110100000000001
Vr20
Vr21
(b)
サイクル
第1スロット
第2スロット
第3スロット
N+1
st
0101010100000010000011
Vr16
Vr17
mod
11001000110100000000001
Vr20
Vr21
mul1
10001000110001000010000001
Vr18
Vr19
(c)
サイクル
第1スロット
第2スロット
第3スロット
N+1
mul1
10001000110001000010000001
Vr18
Vr19
st
0101010100000010000011
Vr16
Vr17
mod
11001000110100000000001
Vr20
Vr21
(d)
サイクル
第1スロット
第2スロット
第3スロット
N+1
mul1
10001000110001000010000001
Vr18
Vr19
mod
11001000110100000000001
Vr20
Vr21
st
0101010100000010000011
Vr16
Vr17
(e)
サイクル
第1スロット
第2スロット
第3スロット
N+1
mod
11001000110100000000001
Vr20
Vr21
st
0101010100000010000011
Vr16
Vr17
mul1
10001000110001000010000001
Vr18
Vr19
(f)
サイクル
第1スロット
第2スロット
第3スロット
N+1
mod
11001000110100000000001
Vr20
Vr21
mul1
10001000110001000010000001
Vr18
Vr19
st
0101010100000010000011
Vr16
Vr17

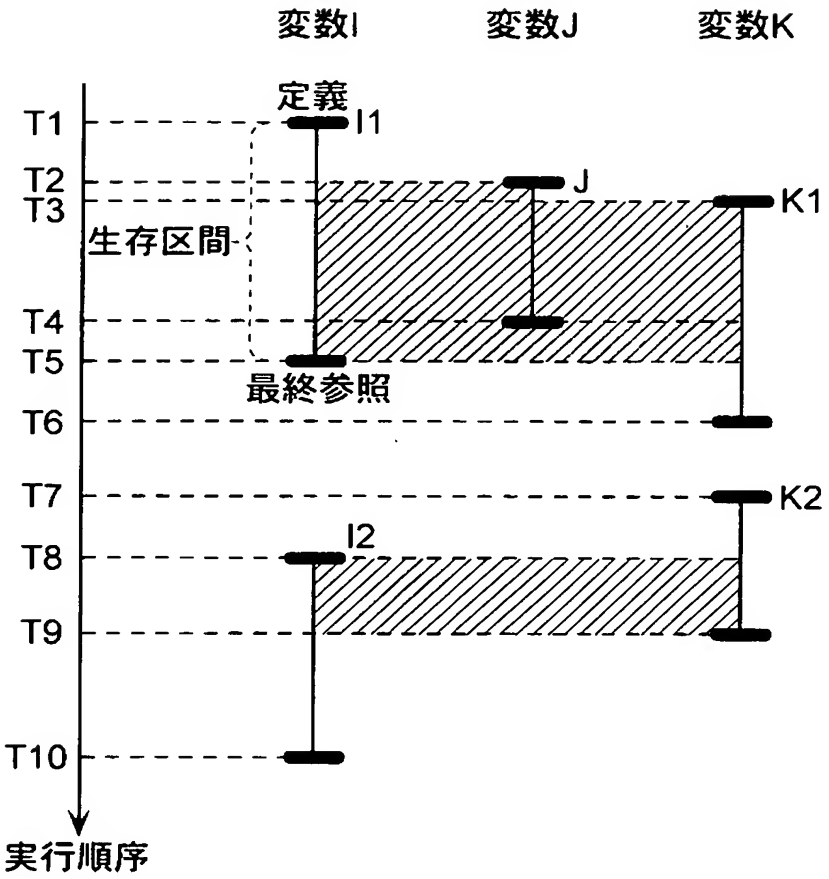
【図２３】


サイクル
第1スロット
第2スロット
第3スロット
N
ld
0101011000000100001
Vr10
Vr11
sub1
1100000001000101010000 1
Vr12
Vr13
add1
1100000000010110000001
Vr14
Vr15
N+1
mul1
1000100011000100000001
Vr18
Vr19
st
0101010100000010000011
Vr16
Vr17
mod
1100100011010000000001
Vr20
Vr21
オペコードの
ハミング距離
10
オペコードの
ハミング距離
9
オペコードの
ハミング距離
5

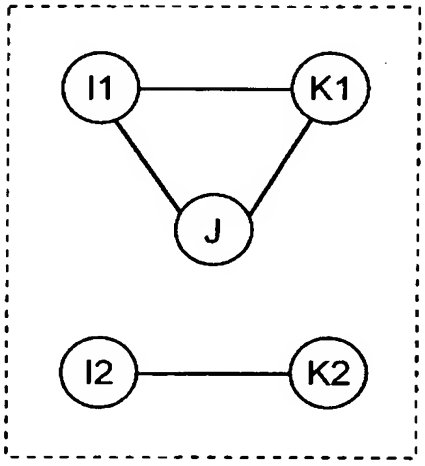
【図２４】


レジスタ割付処理開始
各割付対象(変数)の
生存区間および優先度の算出
S72
干渉グラフを生成
S74
(ループ1)
すべての割付対象について
繰返す
S78
最高優先度の割付対象選択
S80
S82
先行サイクルの
レジスタと同一番号のレジスタ
が割付可能か？
NO
YES
S84
同一番号のレジスタを割付け
S86
先行サイクルの
レジスタ番号と
の最小ハミング
距離を有する割
付可能レジスタ
の特定
S88
2つ以上
ある？
NO
YES
S90
割付可能レジスタ
のうちのいずれか
を任意に選択し、
割付け
S92
割付可能
レジスタ
の割付け
(ループ1)
S94
終了

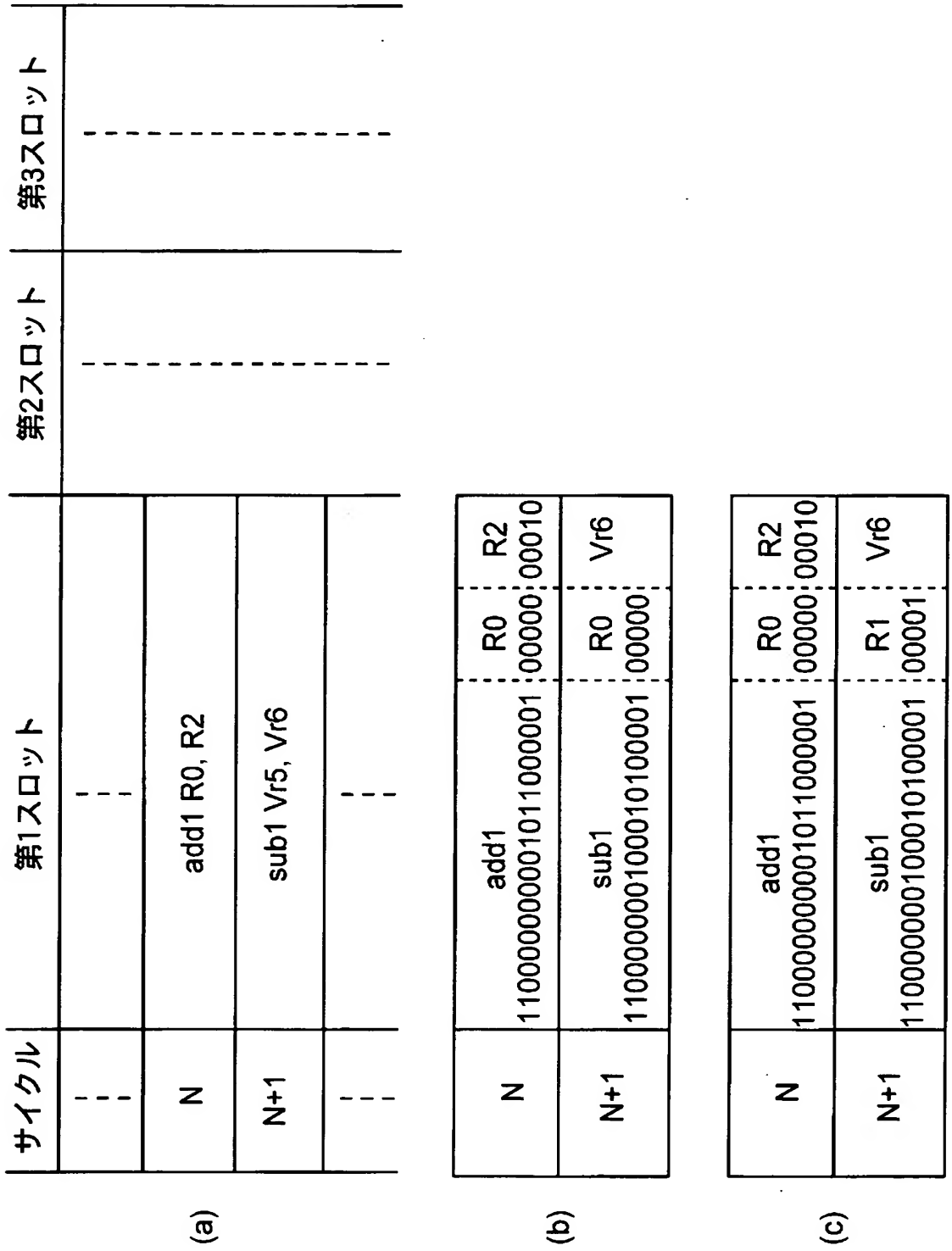
【図２５】


変数I
変数J
変数K
定義
I1
J
K1
T1
T2
T3
生存区間
T4
T5
最終参照
T6
T7
K2
I2
T8
T9
T10
実行順序

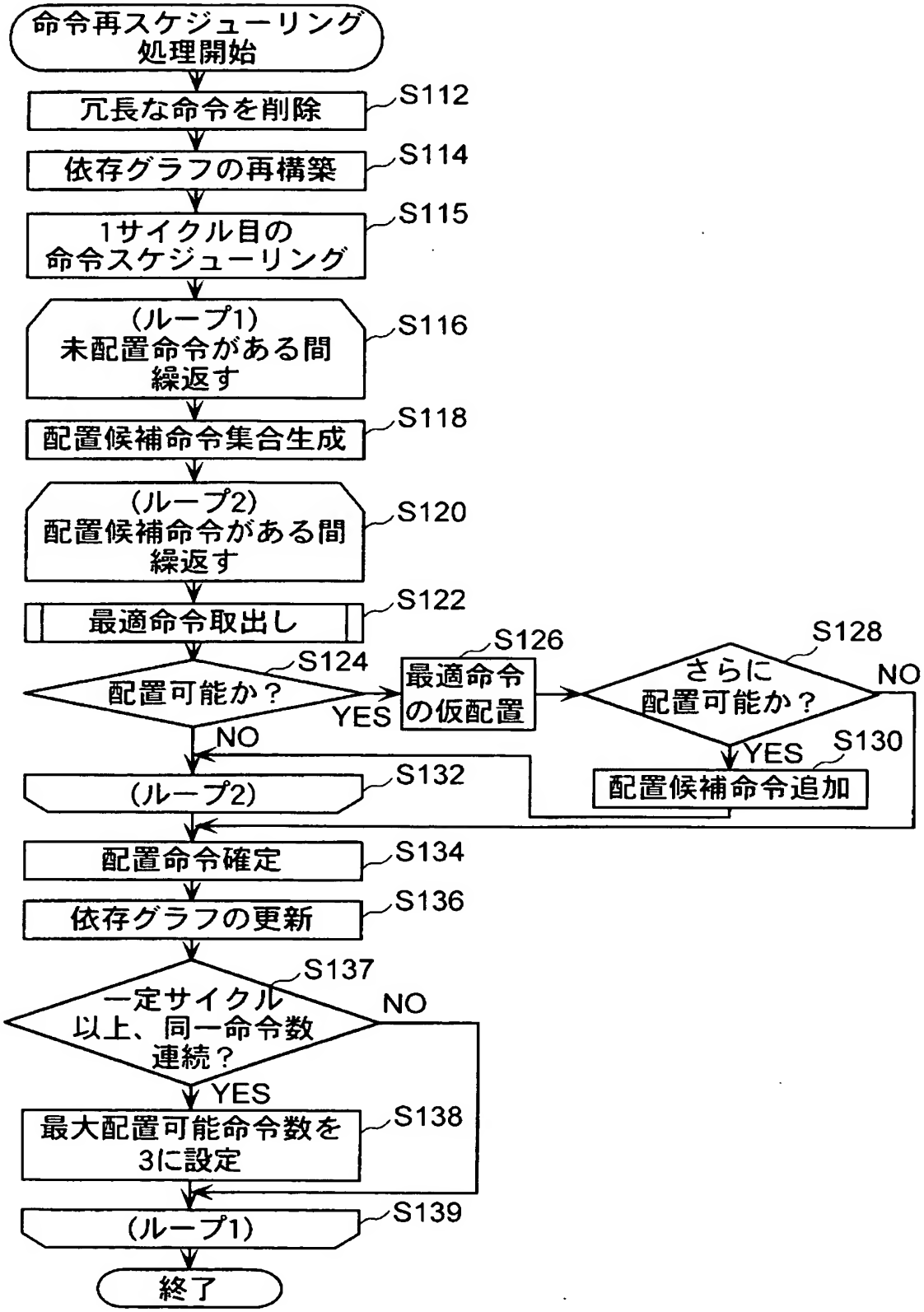
【図２６】


I1
K1
J
I2
K2

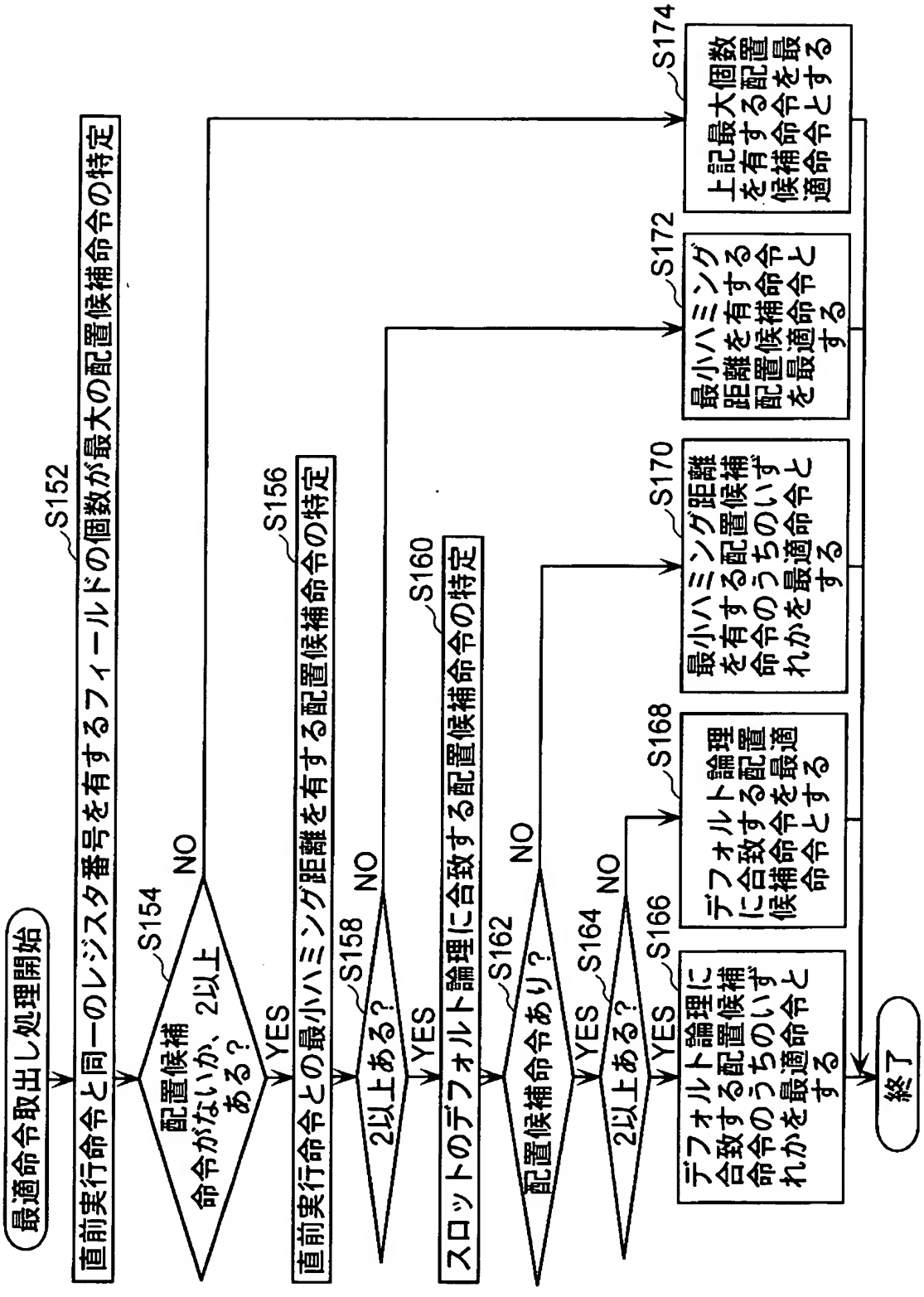
【図２７】


サイクル
第1スロット
第2スロット
第3スロット
N
N+1
add1 R0, R2
sub1 Vr5, Vr6
add1 11000000001011000001 R0 00000 R2 00010
sub1 11000000010001010000 1 R0 00000 Vr6
add1 11000000001011000001 R0 00000 R2 00010
sub1 11000000010001010000 1 R1 00001 Vr6
(a)
(b)
(c)

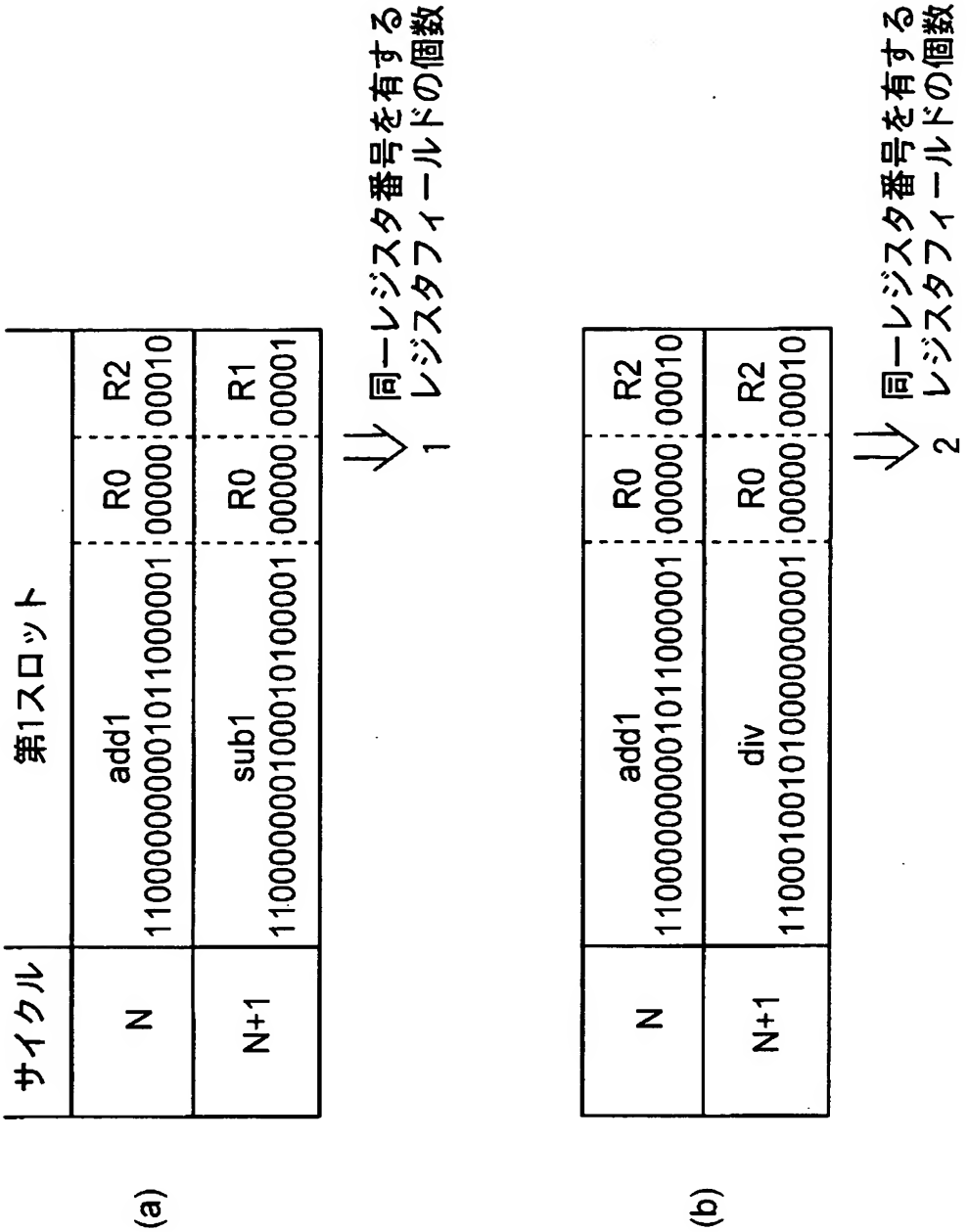
【図２８】


命令再スケジューリング
処理開始
冗長な命令を削除
S112
依存グラフの再構築
S114
1サイクル目の
命令スケジューリング
S115
(ループ1)
未配置命令がある間
繰返す
S116
配置候補命令集合生成
S118
(ループ2)
配置候補命令がある間
繰返す
S120
最適命令取出し
S122
配置可能か？
S124
YES
最適命令
の仮配置
S126
さらに
配置可能か？
S128
NO
YES
S130
配置候補命令追加
NO
(ループ2)
S132
配置命令確定
S134
依存グラフの更新
S136
一定サイクル
以上、同一命令数
連続？
S137
NO
YES
最大配置可能命令数を
3に設定
S138
(ループ1)
S139
終了

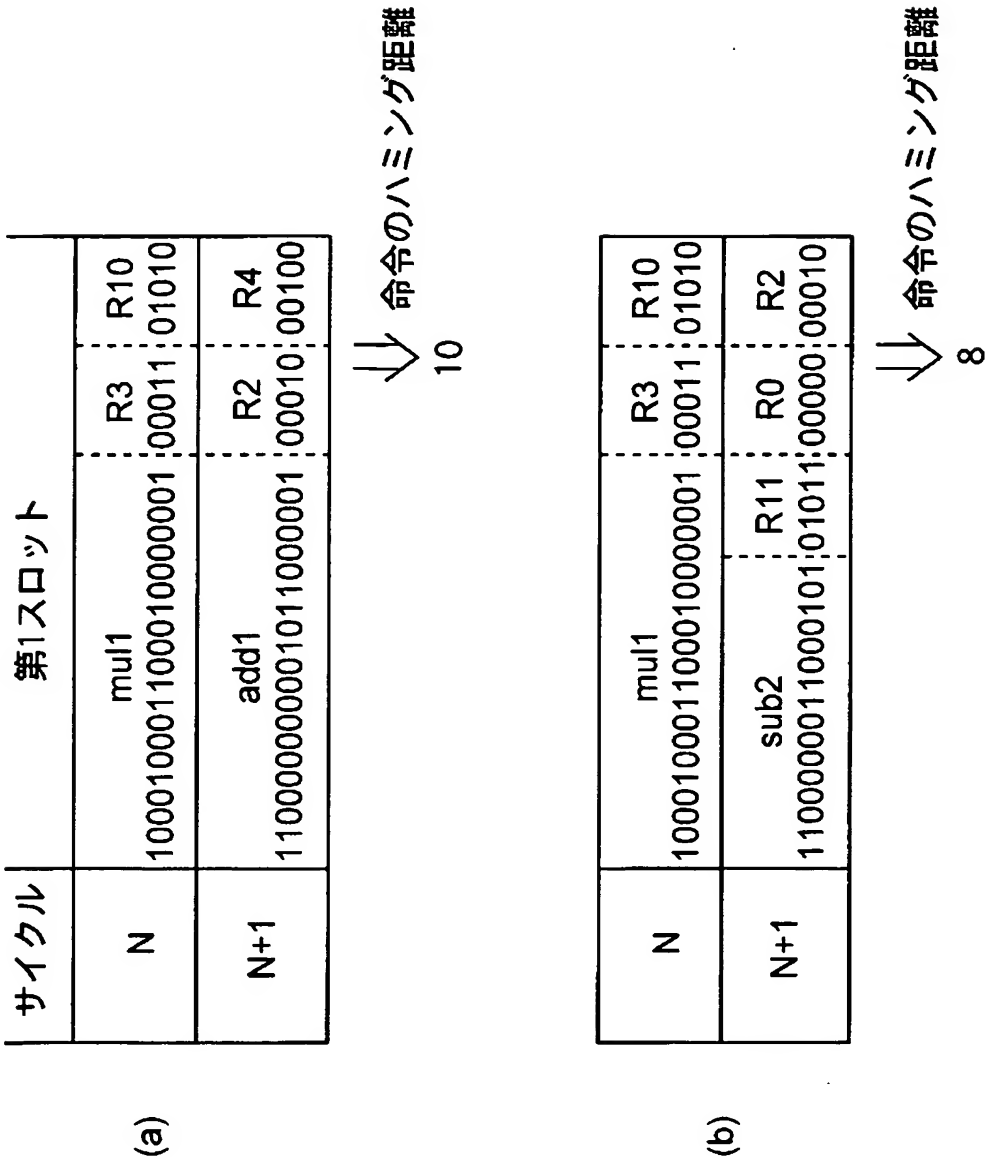
【図２９】


最適命令取出し処理開始
S152
直前実行命令と同一のレジスタ番号を有するフィールドの個数が最大の配置候補命令の特定
S154
配置候補命令がないか、2以上ある？
NO
YES
S156
直前実行命令との最小ハミング距離を有する配置候補命令の特定
S158
2以上ある？
NO
YES
S160
スロットのデフォルト論理に合致する配置候補命令の特定
S162
配置候補命令あり？
NO
YES
S164
2以上ある？
NO
YES
S166
デフォルト論理に合致する配置候補命令のうちのいずれかを最適命令とする
S168
デフォルト論理に合致する配置候補命令を最適命令とする
S170
最小ハミング距離を有する配置候補命令のうちのいずれかを最適命令とする
S172
最小ハミング距離を有する配置候補命令を最適命令とする
S174
上記最大個数を有する配置候補命令を最適命令とする
終了

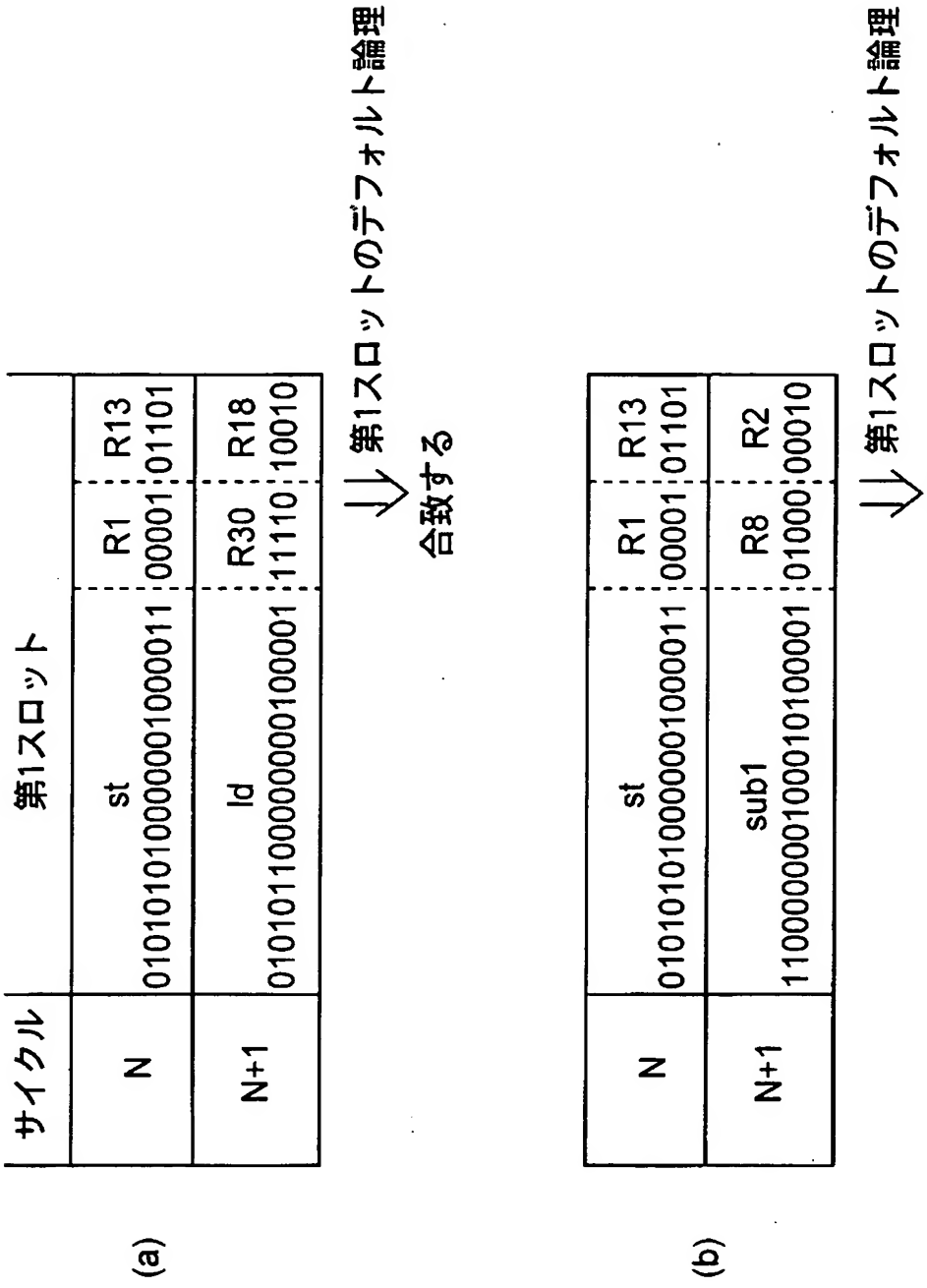
【図３０】


(a)
サイクル
第1スロット
N
add1
11000000000101011000001
R0
00000
R2
00010
N+1
sub1
11000000010001010100001
R0
00000
R1
00001
同一レジスタ番号を有する
レジスタフィールドの個数
1
(b)
サイクル
第1スロット
N
add1
11000000000101011000001
R0
00000
R2
00010
N+1
div
11000100101010000000001
R0
00000
R2
00010
同一レジスタ番号を有する
レジスタフィールドの個数
2

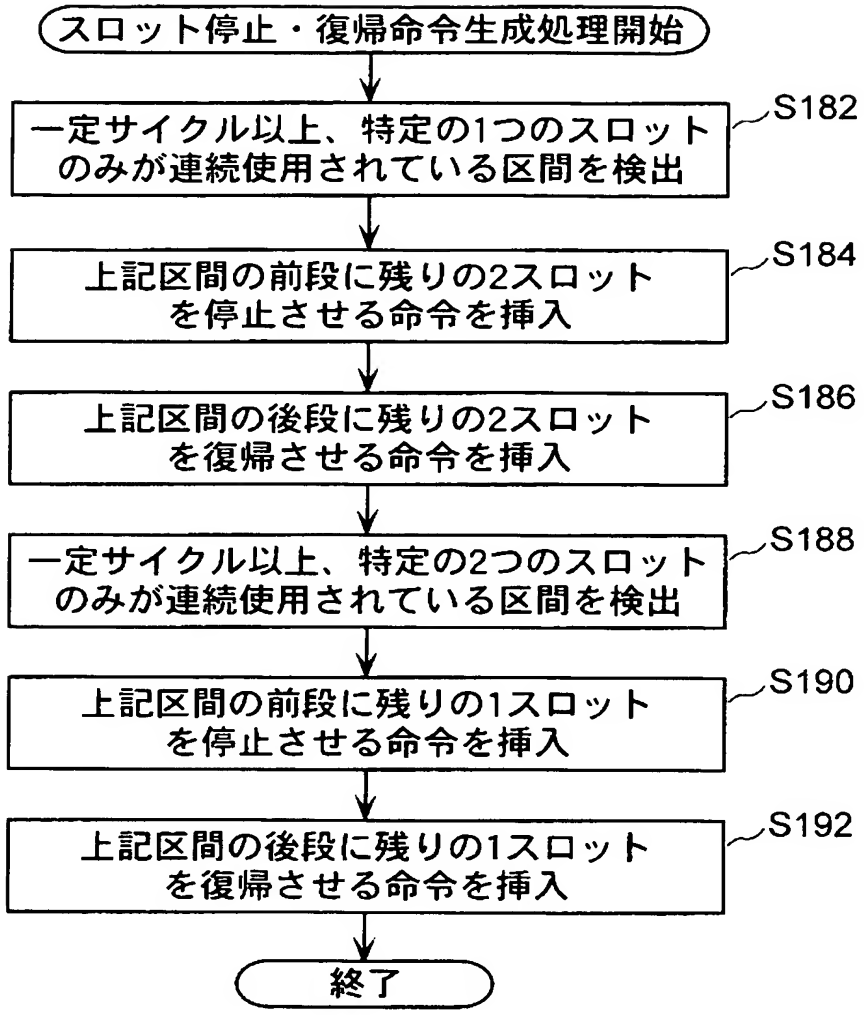
【図３１】


サイクル
第1スロット
N
mul1
1000100011000010000001
R3
00011
R10
01010
N+1
add1
1100000000010110000001
R2
00010
R4
00100
命令のハミング距離
10
(a)
N
mul1
1000100011000010000001
R3
00011
R10
01010
N+1
sub2
1100000011000101
R11
01011
R0
00000
R2
00010
命令のハミング距離
8
(b)

【図３２】


(a)
サイクル
第1スロット
N
st
0101010100000010000011
R1 00001
R13 01101
N+1
ld
0101011000000000100001
R30 11110
R18 10010
第1スロットのデフォルト論理
合致する
(b)
サイクル
第1スロット
N
st
0101010100000010000011
R1 00001
R13 01101
N+1
sub1
1100000001000101010000 1
R8 01000
R2 00010
第1スロットのデフォルト論理
合致しない

【図３３】


スロット停止・復帰命令生成処理開始
一定サイクル以上、特定の1つのスロットのみが連続使用されている区間を検出
S182
上記区間の前段に残りの2スロットを停止させる命令を挿入
S184
上記区間の後段に残りの2スロットを復帰させる命令を挿入
S186
一定サイクル以上、特定の2つのスロットのみが連続使用されている区間を検出
S188
上記区間の前段に残りの1スロットを停止させる命令を挿入
S190
上記区間の後段に残りの1スロットを復帰させる命令を挿入
S192
終了

【図３４】

サイクル	第1スロット	第2スロット	第3スロット
1	▨	▨	▨
2	▨	▨	▨
3	▨	▨	▨
4	▨	▨	
5	▨	▨	
6	▨	▨	
7	▨	▨	
8	▨	▨	
9	▨	▨	
10	▨		
11	▨		
12	▨		
13	▨		
14	▨		
15	▨		
16	▨		
17	▨		
18	▨		
19	▨	▨	
20	▨	▨	▨

9サイクル

▨：命令配置済み

【図３５】

サイクル	第1スロット	第2スロット	第3スロット
1			
2			
3			
4			
5			
6			
7			
8			
9			set1 1
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			clear1 1
20			

5サイクル

9サイクル

：命令配置済み

【図３６】

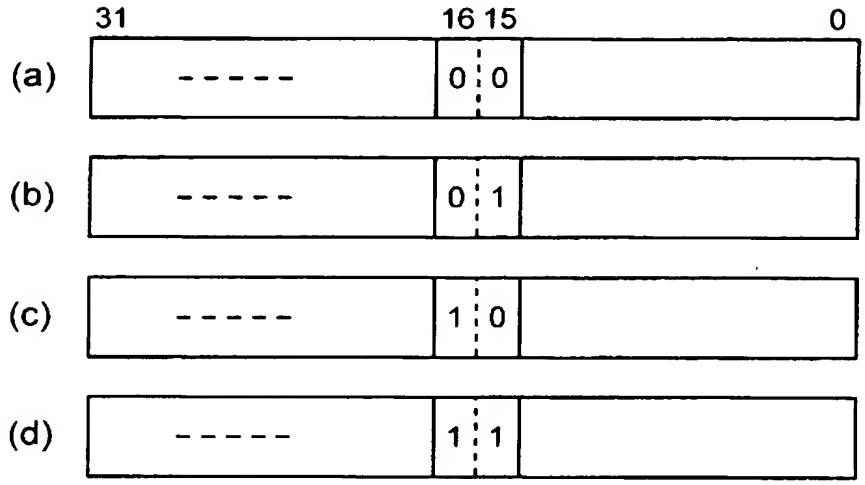
サイクル	第1スロット	第2スロット	第3スロット
1			
2			
3			
4	set2 12		
5			
6			
7			
8			
9			
10	clear2 12		
11			set1 1
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			clear1 1
22			

5サイクル

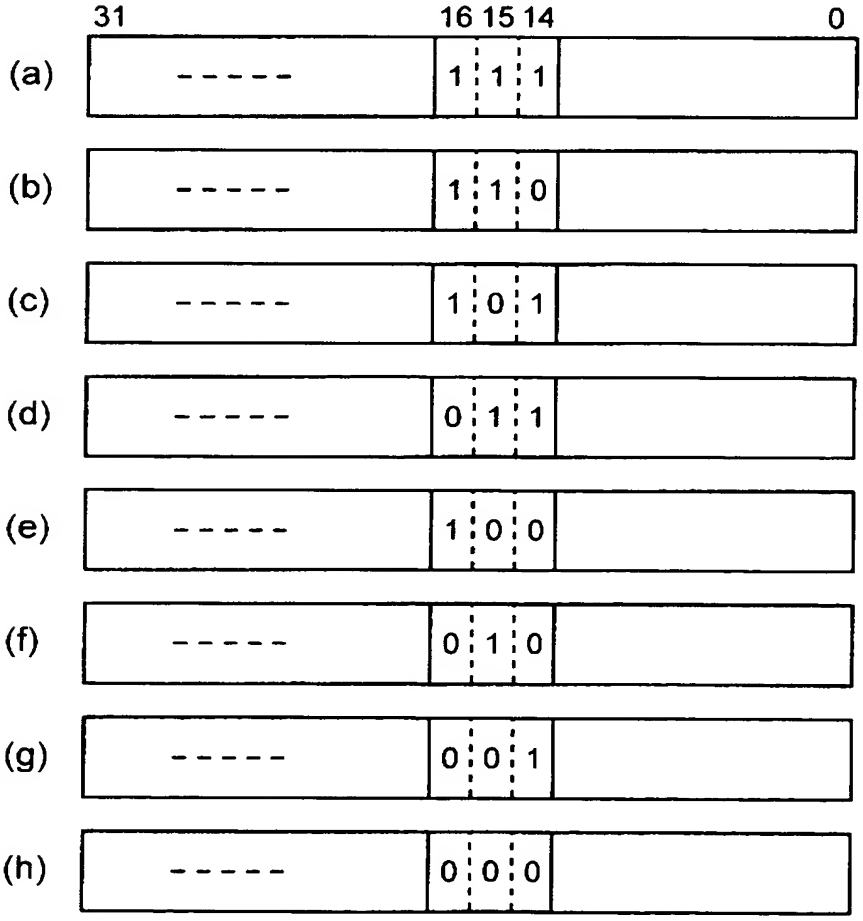
9サイクル

：命令配置済み

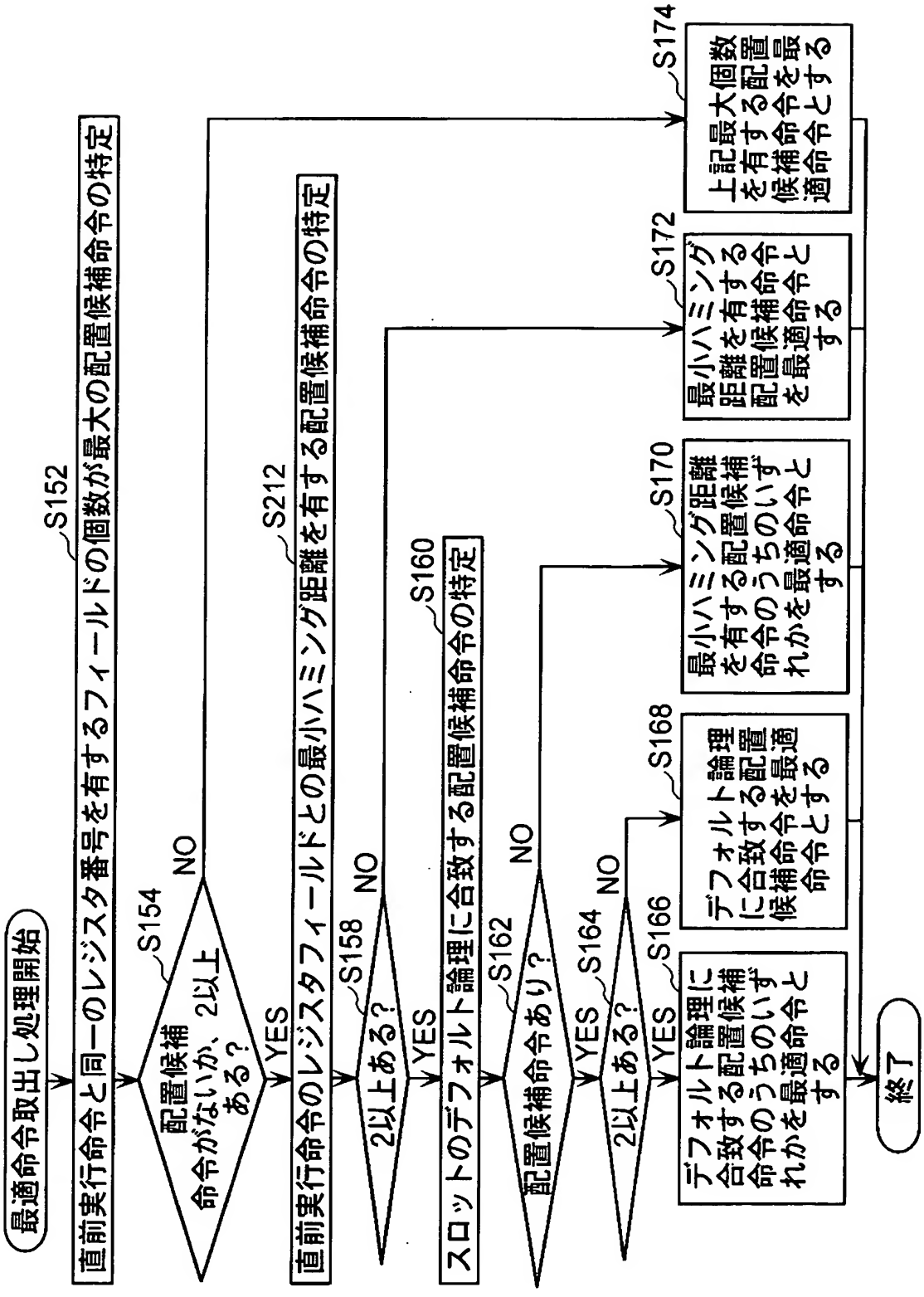
【図３７】


31
16 15
0
(a)
- - - - -
0 0
(b)
- - - - -
0 1
(c)
- - - - -
1 0
(d)
- - - - -
1 1

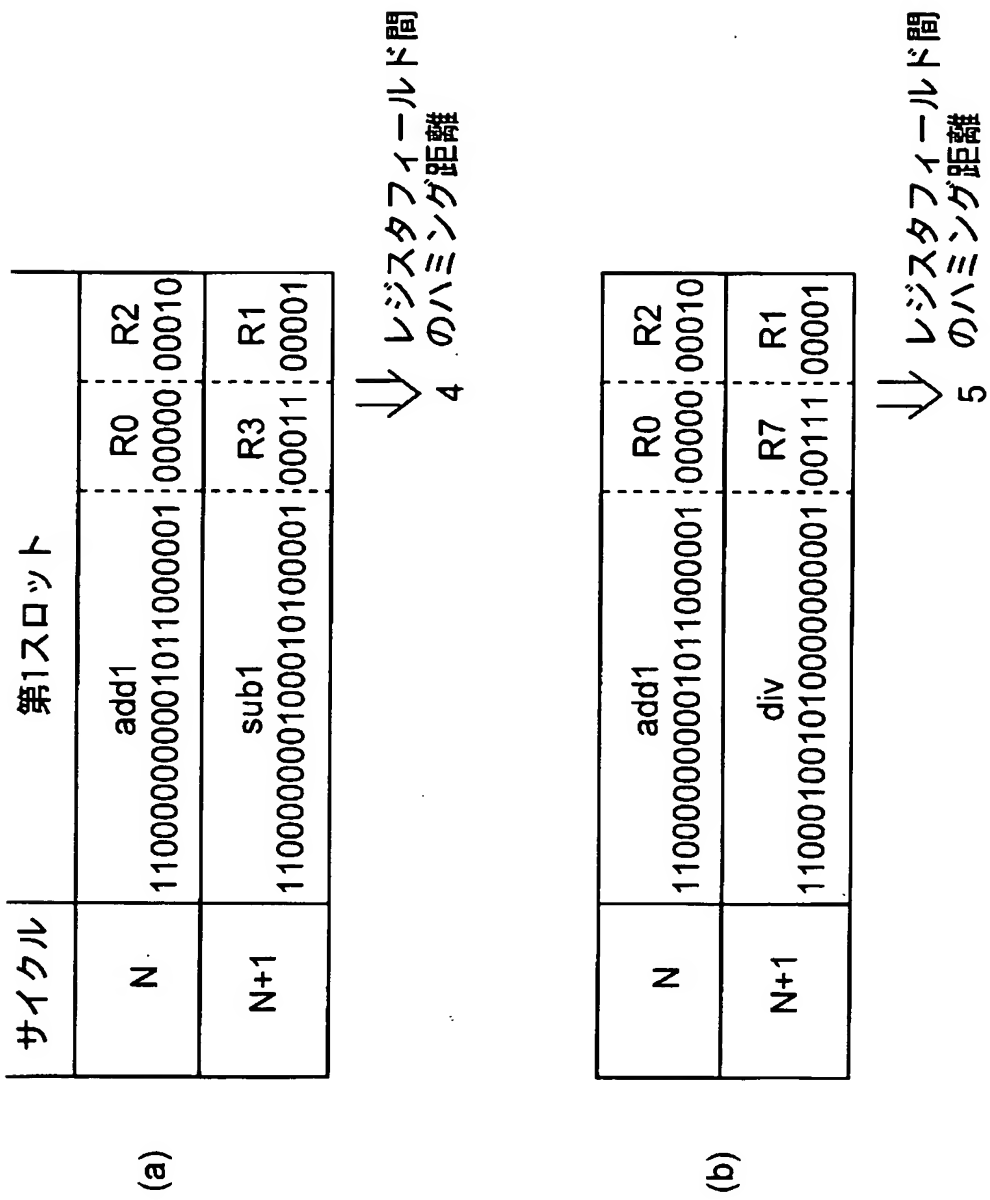
【図３８】


31
16 15 14
0
(a)
- - - - -
1 1 1
(b)
- - - - -
1 1 0
(c)
- - - - -
1 0 1
(d)
- - - - -
0 1 1
(e)
- - - - -
1 0 0
(f)
- - - - -
0 1 0
(g)
- - - - -
0 0 1
(h)
- - - - -
0 0 0

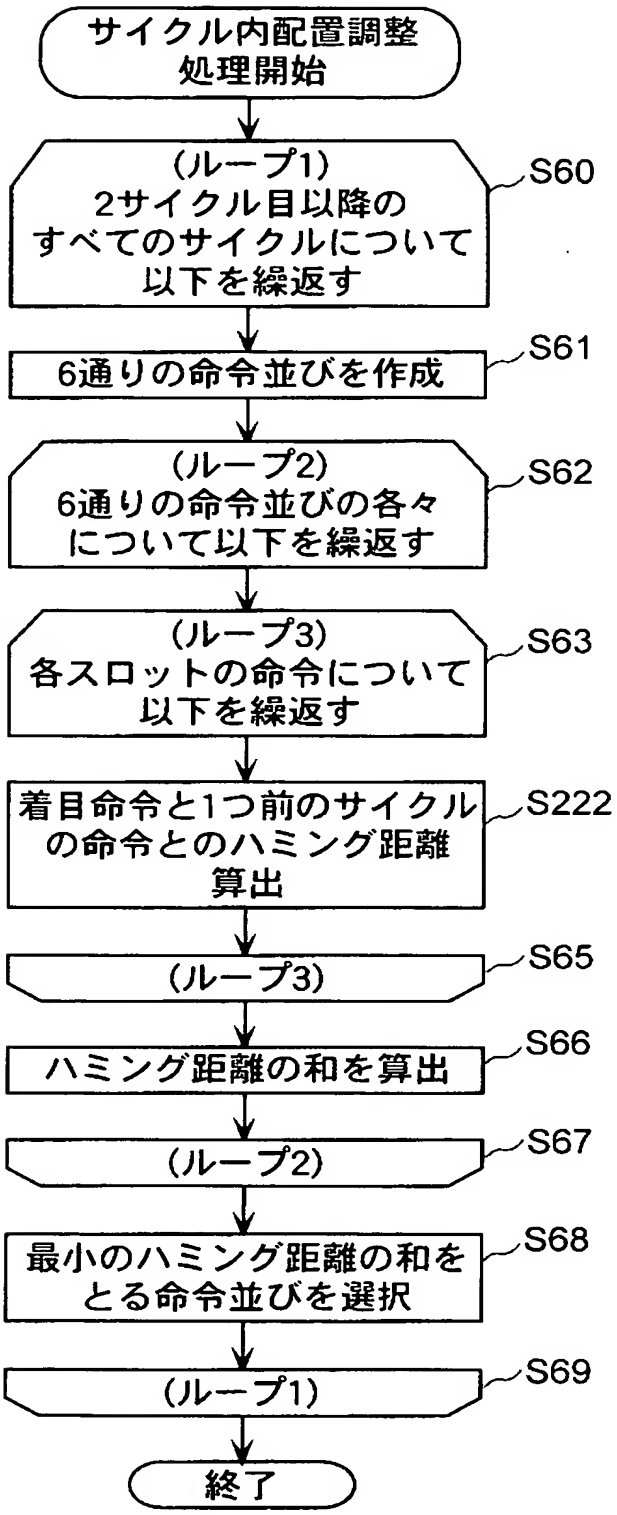
【図３９】


最適命令取出し処理開始
S152
直前実行命令と同一のレジスタ番号を有するフィールドの個数が最大の配置候補命令の特定
S154
配置候補命令がないか、2以上ある？
NO
YES
S212
直前実行命令のレジスタフィールドとの最小ハミング距離を有する配置候補命令の特定
S158
2以上ある？
NO
YES
S160
スロットのデフォルト論理に合致する配置候補命令の特定
S162
配置候補命令あり？
NO
YES
S164
2以上ある？
NO
YES
S166
デフォルト論理に合致する配置候補命令のうちのいずれかを最適命令とする
S168
デフォルト論理に合致する配置候補命令を最適命令とする
S170
最小ハミング距離を有する配置候補命令のうちのいずれかを最適命令とする
S172
最小ハミング距離を有する配置候補命令を最適命令とする
S174
上記最大個数を有する配置候補命令を最適命令とする
終了

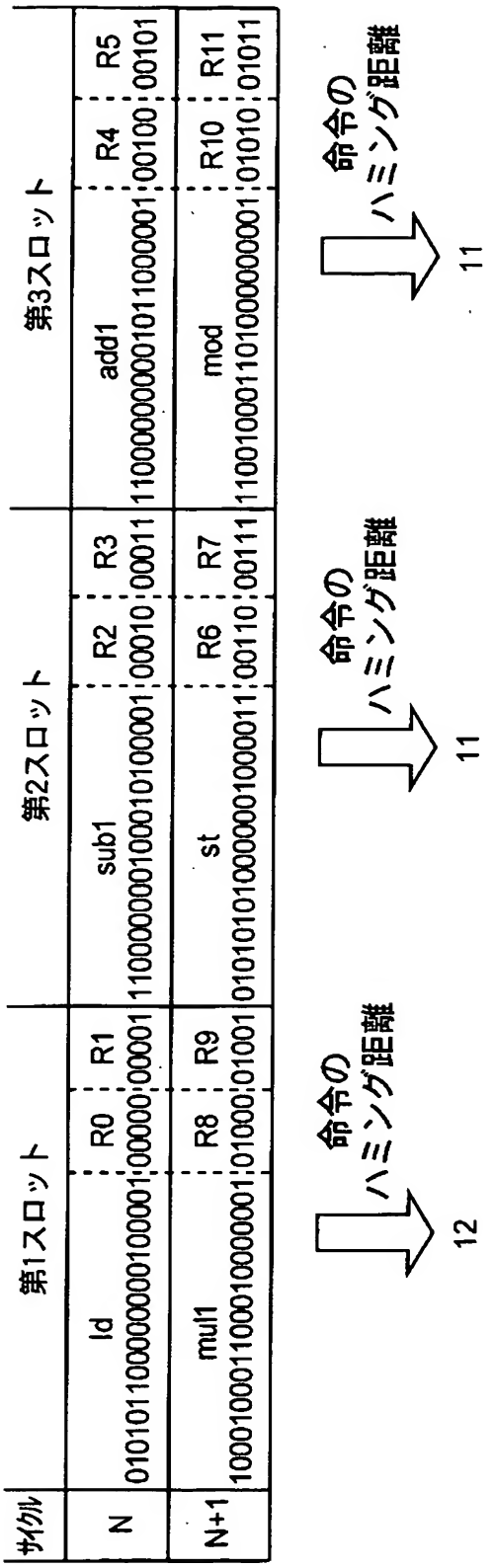
【図４０】


(a)
サイクル
第1スロット
N
add1
11000000000101100000 1
R0
00000
R2
00010
N+1
sub1
11000000010001010000 1
R3
00011
R1
00001
レジスタフィールド間
4 のハミング距離
(b)
サイクル
第1スロット
N
add1
11000000000101100000 1
R0
00000
R2
00010
N+1
div
11000100101010000000 1
R7
00111
R1
00001
レジスタフィールド間
5 のハミング距離

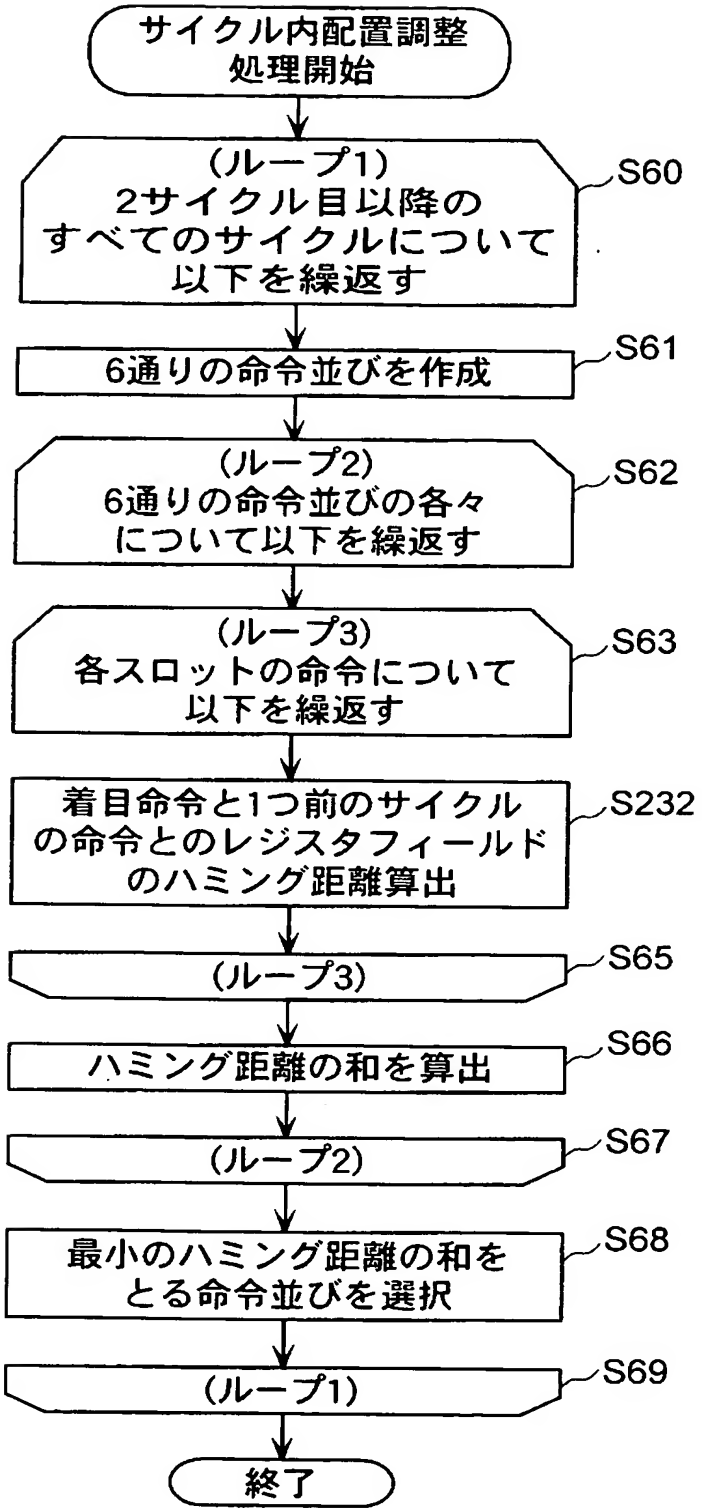
【図４１】


サイクル内配置調整
処理開始
(ループ1)
2サイクル目以降の
すべてのサイクルについて
以下を繰返す
S60
6通りの命令並びを作成
S61
(ループ2)
6通りの命令並びの各々
について以下を繰返す
S62
(ループ3)
各スロットの命令について
以下を繰返す
S63
着目命令と1つ前のサイクル
の命令とのハミング距離
算出
S222
(ループ3)
S65
ハミング距離の和を算出
S66
(ループ2)
S67
最小のハミング距離の和を
とる命令並びを選択
S68
(ループ1)
S69
終了

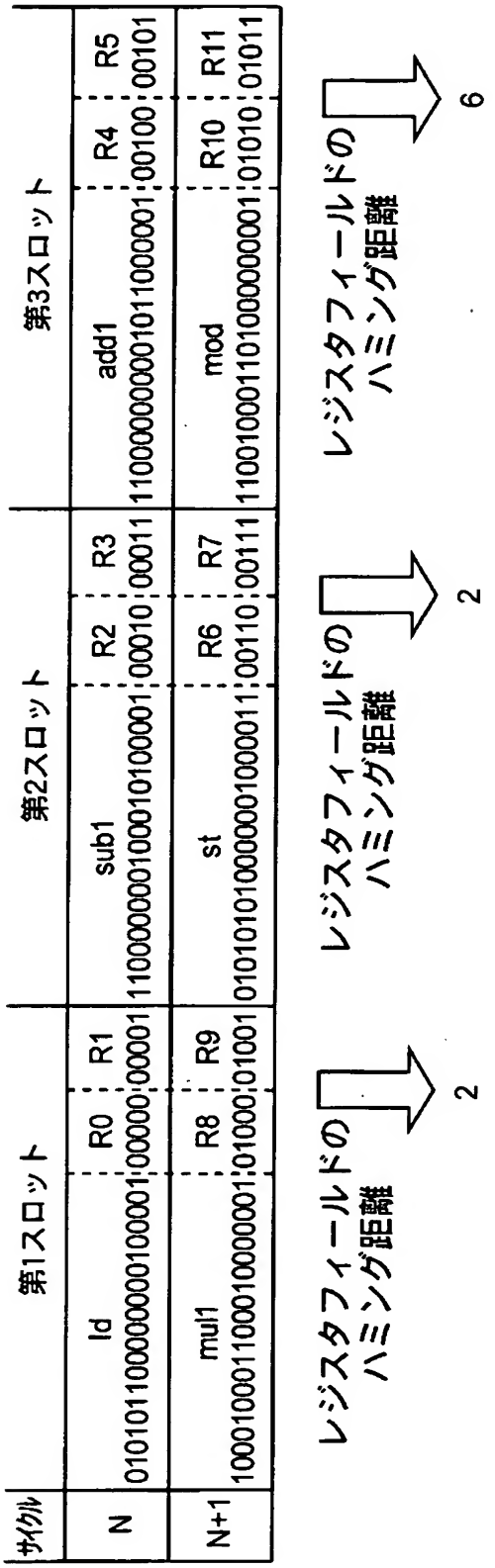
【図４２】


サイクル
第1スロット
第2スロット
第3スロット
N
N+1
ld
0101011000000010000100001
R0 00000
R1 00001
sub1
1100000001000101010000100
R2 00010
R3 00011
add1
1100000000010110000001
R4 00100
R5 00101
mul1
1000100011000100000001
R8 01000
R9 01001
st
0101010100000010000011
R6 00110
R7 00111
mod
1100100011010000000001
R10 01010
R11 01011
命令の
ハミング距離
12
命令の
ハミング距離
11
命令の
ハミング距離
11

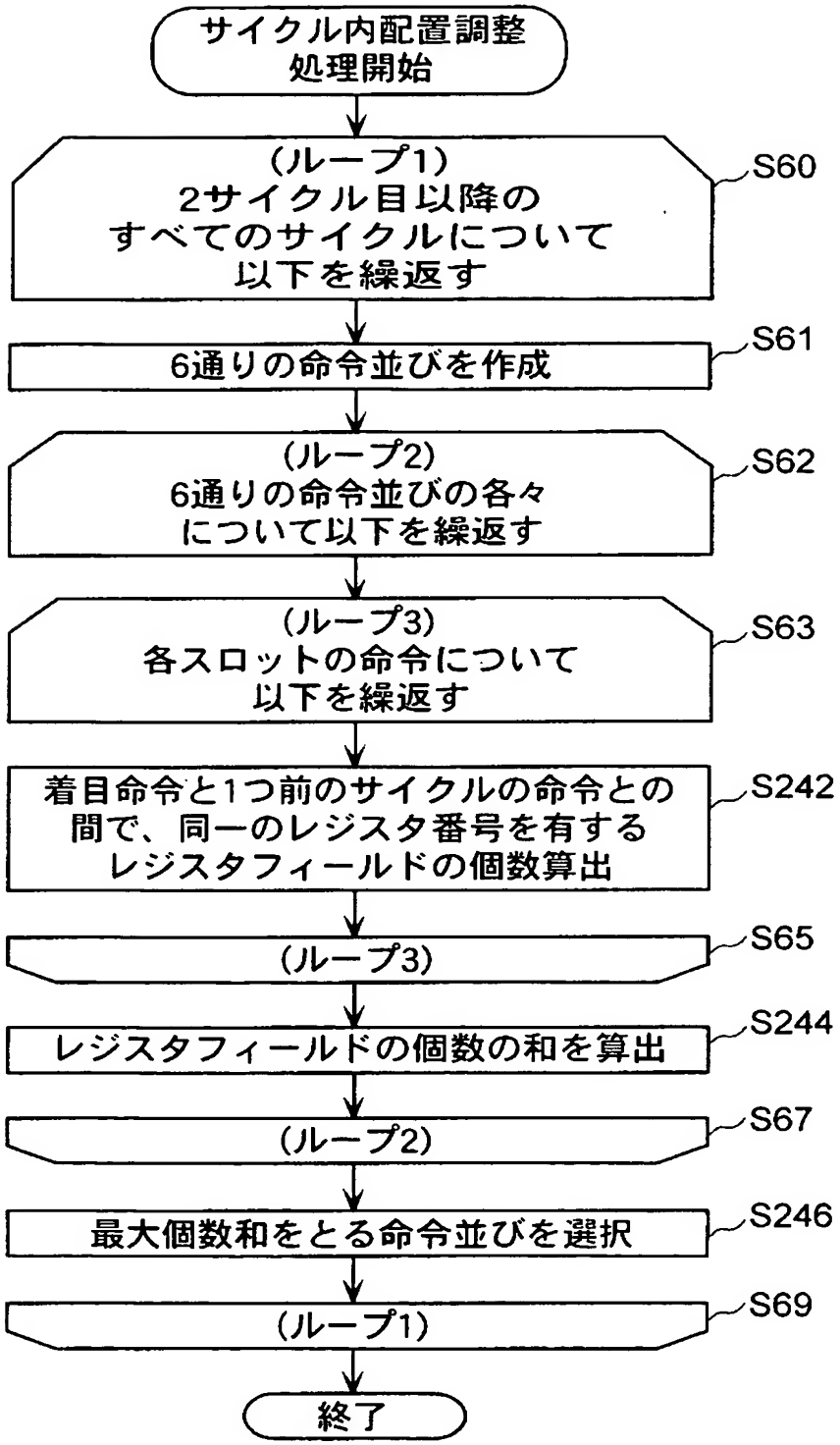
【図４３】


サイクル内配置調整
処理開始
(ループ1)
2サイクル目以降の
すべてのサイクルについて
以下を繰返す
S60
6通りの命令並びを作成
S61
(ループ2)
6通りの命令並びの各々
について以下を繰返す
S62
(ループ3)
各スロットの命令について
以下を繰返す
S63
着目命令と1つ前のサイクル
の命令とのレジスタフィールド
のハミング距離算出
S232
(ループ3)
S65
ハミング距離の和を算出
S66
(ループ2)
S67
最小のハミング距離の和を
とる命令並びを選択
S68
(ループ1)
S69
終了

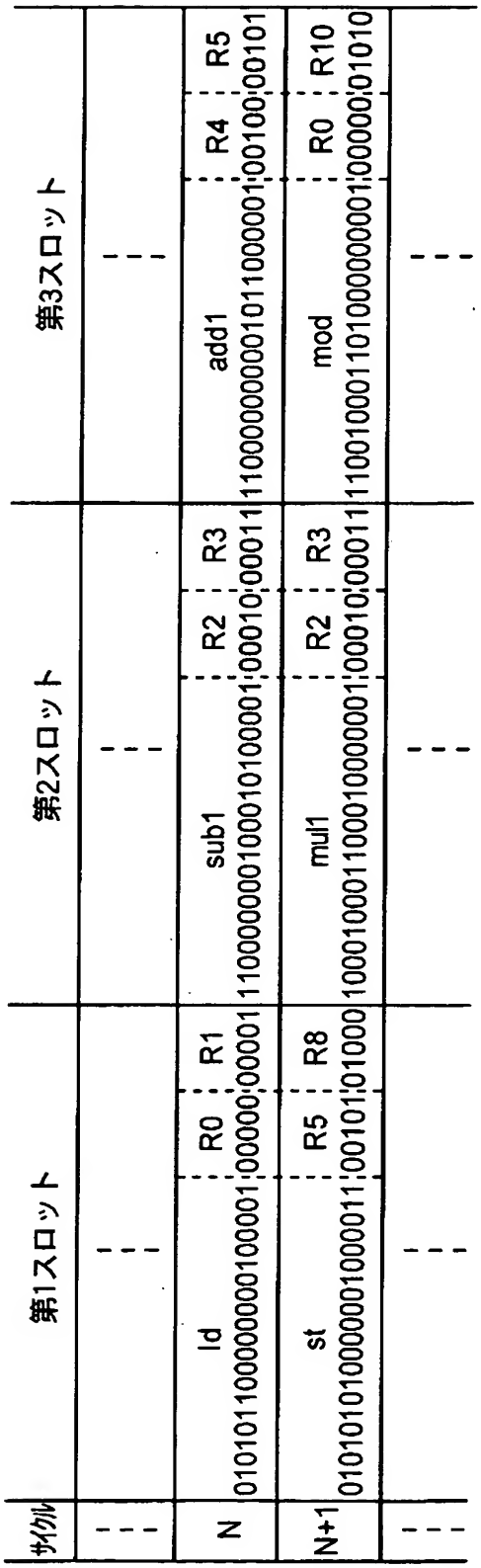
【図４４】


サイクル
第1スロット
第2スロット
第3スロット
N
N+1
ld
R0
R1
sub1
R2
R3
add1
R4
R5
mul1
R8
R9
st
R6
R7
mod
R10
R11
レジスタフィールドの
ハミング距離
2
レジスタフィールドの
ハミング距離
2
レジスタフィールドの
ハミング距離
6

【図４５】


サイクル内配置調整
処理開始
(ループ1)
2サイクル目以降の
すべてのサイクルについて
以下を繰返す
S60
6通りの命令並びを作成
S61
(ループ2)
6通りの命令並びの各々
について以下を繰返す
S62
(ループ3)
各スロットの命令について
以下を繰返す
S63
着目命令と1つ前のサイクルの命令との
間で、同一のレジスタ番号を有する
レジスタフィールドの個数算出
S242
(ループ3)
S65
レジスタフィールドの個数の和を算出
S244
(ループ2)
S67
最大個数和をとる命令並びを選択
S246
(ループ1)
S69
終了

【図４６】


サイクル
第1スロット
第2スロット
第3スロット

N
ld
0101011100000001000011 R0 00000 R1 00001
sub1
1100000001000101010000 1 R2 00010 R3 00011
add1
1100000000010110000001 R4 00100 R5 00101
N+1
st
0101010100000010000011 R5 00101 R8 01000
mul1
1000100011000100000001 R2 00010 R3 00011
mod
1100100011010000000001 R0 00000 R10 01010

【図４７】

(a)

サイクル	第1スロット			第2スロット			第3スロット		
N+1	st 0101010100000010000111	R5 00101	R8 01000	mul1 1000100011000100000001	R2 00010	R3 00011	mod 1100100011010000000001	R0 00000	R10 01010

(b)

サイクル	第1スロット			第2スロット			第3スロット		
N+1	st 0101010100000010000111	R5 00101	R8 01000	mod 1100100011010000000001	R0 00000	R10 01010	mul1 1000100011000100000001	R2 00010	R3 00011

(c)

サイクル	第1スロット			第2スロット			第3スロット		
N+1	mul1 1000100011000100000001	R2 00010	R3 00011	st 0101010100000010000111	R5 00101	R8 01000	mod 1100100011010000000001	R0 00000	R10 01010

(d)

サイクル	第1スロット			第2スロット			第3スロット		
N+1	mul1 1000100011000100000001	R2 00010	R3 00011	mod 1100100011010000000001	R0 00000	R10 01010	st 0101010100000010000111	R5 00101	R8 01000

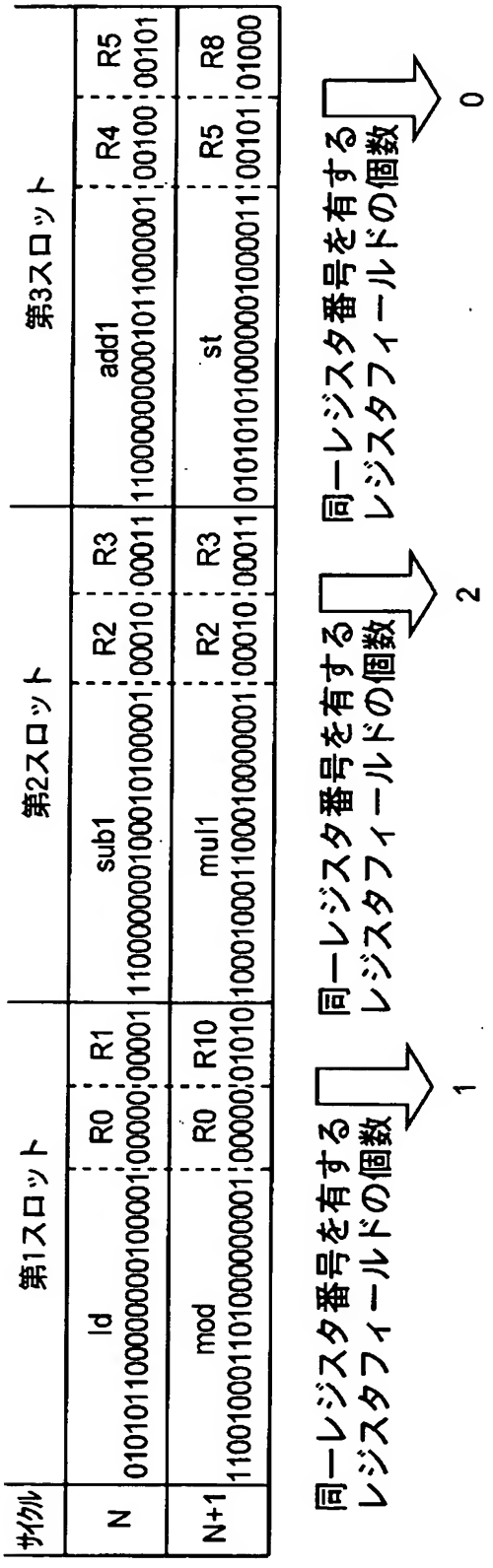
(e)

サイクル	第1スロット			第2スロット			第3スロット		
N+1	mod 1100100011010000000001	R0 00000	R10 01010	st 0101010100000010000111	R5 00101	R8 01000	mul1 1000100011000100000001	R2 00010	R3 00011

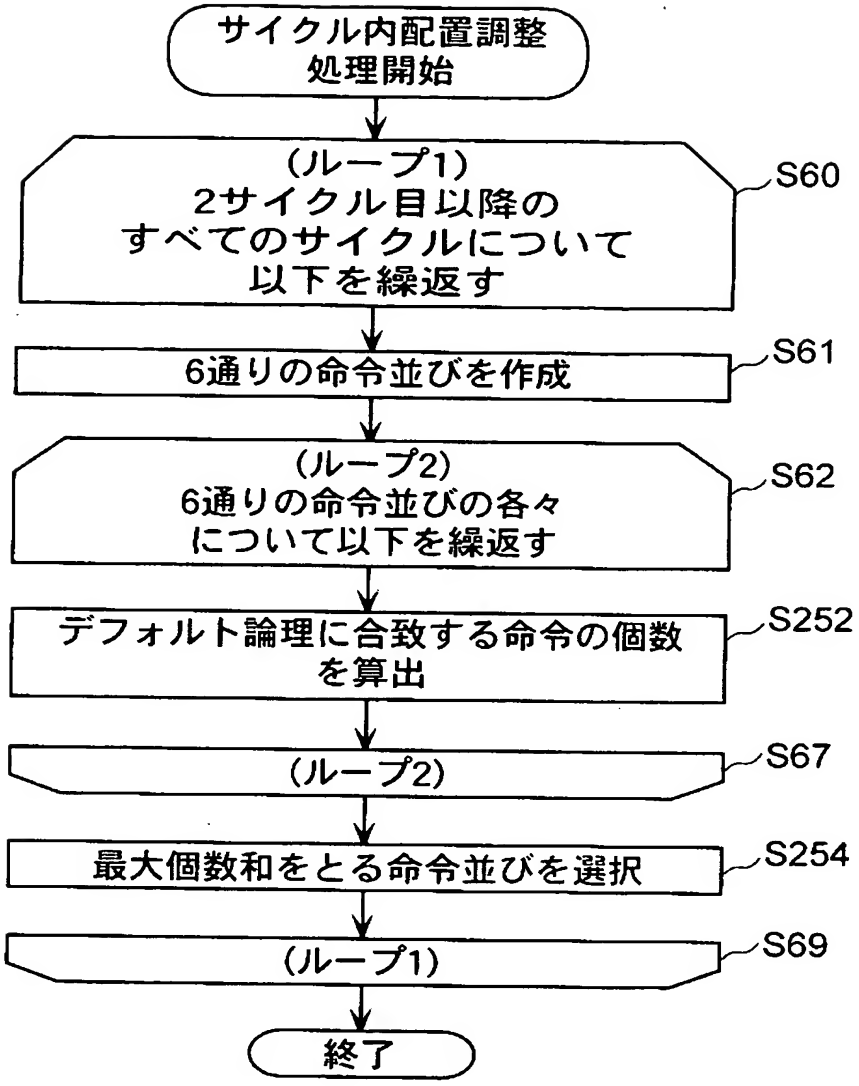
(f)

サイクル	第1スロット			第2スロット			第3スロット		
N+1	mod 1100100011010000000001	R0 00000	R10 01010	mul1 1000100011000100000001	R2 00010	R3 00011	st 0101010100000010000111	R5 00101	R8 01000

【図４８】


サイクル
第1スロット
第2スロット
第3スロット
N
N+1
ld
R0
R1
sub1
R2
R3
add1
R4
R5
mod
R0
R10
mul1
R2
R3
st
R5
R8
同一レジスタ番号を有するレジスタフィールドの個数
1
同一レジスタ番号を有するレジスタフィールドの個数
2
同一レジスタ番号を有するレジスタフィールドの個数
0

【図４９】


サイクル内配置調整
処理開始
(ループ1)
2サイクル目以降の
すべてのサイクルについて
以下を繰返す
S60
6通りの命令並びを作成
S61
(ループ2)
6通りの命令並びの各々
について以下を繰返す
S62
デフォルト論理に合致する命令の個数
を算出
S252
(ループ2)
S67
最大個数和をとる命令並びを選択
S254
(ループ1)
S69
終了

【書類名】　　要約書

【要約】

【課題】　並列処理可能なプロセッサを低消費電力で動作させることができる命令列を生成可能なコンパイラを提供する。

【解決手段】　着目しているサイクルの３つの命令について並べ替えを行ない、６通りの命令並びを作成する（Ｓ６１）。各命令並びの各スロットについて、着目命令と１つ前のサイクルの命令とのオペコードのビットパターン間のハミング距離を求める（Ｓ６４）。その処理を３つのスロットの命令のすべてについて行ない（Ｓ６３～Ｓ６５）、ハミング距離の和を求める（Ｓ６６）。以上の処理を、６通りの命令並びのすべてについて行なう（Ｓ６２～Ｓ６７）。６通りハミング距離の和のうち最小値をとる命令並びを選択し、その命令並びの並びになるように命令の並べ替えを行なう（Ｓ６８）。以上の処理を、２サイクル目から最終サイクルまで繰返す（Ｓ６０～Ｓ６９）。

【選択図】　　図１９

認定・付加情報

特許出願の番号	特願２００３－０１９３６５
受付番号	５０３００１３５０５５
書類名	特許願
担当官	第七担当上席　　００９６
作成日	平成１５年　１月２９日

＜認定情報・付加情報＞

【提出日】　平成15年 1月28日

次頁無

特願２００３－０１９３６５

出　願　人　履　歴　情　報

識別番号　　［０００００５８２１］

１．変更年月日　　１９９０年　８月２８日
［変更理由］　　新規登録
住　所　　大阪府門真市大字門真１００６番地
氏　名　　松下電器産業株式会社